THE LIB SHELL COMMAND REFERENCE MANUAL

Manual No. 10131-MF-00
September 18, 1989
Revision 7

matrox
electronic systems

THE LIB SHELL
COMMAND REFERENCE
MANUAL

Manual No. 10131-MF-00
September 18, 1989
Revision 7

This Manual Is Valid For The Following Products:	
Name	Revision
PG-Series-1 LIB Shell User Software	03.20 and up
PG2-Series-1 LIB Shell User Software	03.20 and up
VG-Series-1 LIB Shell User Software	03.20 and up
MG-Series-1 LIB Shell User Software	03.20 and up
MMG-Series-1 LIB Shell User Software	03.20 and up

MATROX Electronic Systems Limited,
1055 St. Regis Boulevard,
Dorval, Quebec,
CANADA H9P 2T4

Telephone: (514)685-2630 Telex: 05-822798
FAX: (514)685-2853

Trademark Acknowledgements

- CGA and EGA are a trademarks of International Business Machines Corp.
- MS-DOS and XENIX are trademarks of Microsoft Corp.
- UNIX is a trademark of AT&T Bell Laboratories.

Contents

1 Introduction **1 – 1**

1.1 Introduction . . . . . 1 – 2

2 Type Definitions and Data Structures **2 – 1**

2.1 Introduction . . . . . 2 – 2

2.2 Short Integers . . . . . 2 – 3

2.3 Long Integers . . . . . 2 – 3

2.4 FIXPT and Floating Point Numbers . . . . . 2 – 4

2.5 Character Strings . . . . . 2 – 5

2.6 Screen Coordinates . . . . . 2 – 5

2.7 2D Virtual Coordinates . . . . . 2 – 6

2.8 3D Virtual Coordinates . . . . . 2 – 7

2.9 Color Intensities . . . . . 2 – 8

2.10 Graphics Cursor Drawing Attributes . . . . . 2 – 8

2.11 Bit Maps . . . . . 2 – 12

2.12 Font Specification . . . . . 2 – 14

2.12.1 Text Offset Table Entry . . . . . 2 – 18

2.12.2 Kerned Pair Entry . . . . . 2 – 19

3 Command Descriptions **3 – 1**

A Default Parameter Values **A – 1**

A.1 Default System Parameters . . . . . A – 2

A.2 Default LUT Contents . . . . . A – 5

B Drawing Attributes **B – 1**

C Summary of Commands **C – 1**

C.1 Transforms . . . . . C – 2

C.2 LUTs . . . . . C – 4

C.3 The Drawing Pen . . . . . C – 4

C.4 The Graphics Cursor . . . . . C – 5

C.5 Outlines . . . . . C – 5

C.6 Fills . . . . . C – 8

C.7 Text . . . . . C – 10

C.8 Command Lists . . . . . C – 12

C.9 Drawing Attributes . . . . . C – 12

C.10 Object and Layer Tagging . . . . . C – 13

C.11 Primitive Raster Operations . . . . . C – 13

C.12 Picking . . . . . C – 14

C.13 Bit Map Storage Areas . . . . . C – 15

C.14 System Inquiries . . . . . C – 15

C.15 Miscellaneous . . . . . C – 16

D LIB Shell Opcodes **D – 1**

D.1 Opcodes in Numerical Order . . . . . D – 2

D.2 Opcodes by Command . . . . . D – 8

Chapter 1

Introduction

SECTION 1.1 Introduction

1.1 Introduction

This is *The LIB Shell Command Reference Manual* which supplements *The LIB Shell Programmer's Manual.* It is intended to be a quick reference for those users already familiar with the LIB Shell.

In this manual we describe the LIB Shell commands, the LSCLIB C language binding and the TMLS Interpreter. Chapters devoted to data types, default parameters, and other useful information are also included.

The LIB Shell is a command decoder which runs on several Matrox graphics boards, including: the PG-Series-1, the PG2-Series-1, the MG-Series-1 , MMG-Series-1, and the VG-Series-1. It provides computer graphics programmers and application developers with a wide range of both standard and specialized graphics functions.

This manual is organized as follows:

Chapter 2 : **Type Definitions and Data Structures.** Type definitions and data structures required by LIB Shell routines, their C language bindings, and corresponding TMLS Interpreter commands.

Chapter 3 : **Command Descriptions.** Detailed descriptions of the LIB Shell commands and their equivalent C language binding and TMLS Interpreter commands, including formats, parameters, and returned values.

Appendix A : **Default Parameter Values.** The default parameter values of the LIB Shell commands.

Appendix B : **Drawing Attributes.** The drawing attributes summarized with respect to the different modules.

Appendix C : **Summary of Commands.** Brief descriptions of the LIB Shell commands by command type.

Appendix D : **LIB Shell Opcodes.** The LIB Shell opcodes in order of opcode and name.

INTRODUCTION

Your documentation package should also include a copy of the product-specific hardware installation manual and LIB Shell supplemental manual:

- PG-Series-1 and SM-Series-1

  The PG/SM-Series-1 LIB Shell Supplemental Manual

  The PG/SM-Series-1 Installation & Technical Manual

- PG2-1281

  The PG2-1281 LIB Shell Supplemental Manual

  The PG2-1281 Installation & Technical Manual

- VG-Series-1

  The VG-Series-1 LIB Shell Supplemental Manual

  The VG-Series-1 Installation & Technical Manual

- MMG-Series-1

  The MMG-Series-1 Installation & Supplemental Manual

- MG-Series-1

  The MG-Series-1 LIB Shell Supplemental Manual

  The MG-Series-1 Installation & Technical Manual

The LIB Shell software, utility programs, and demonstration programs are on your LIB Shell User Software Diskette. Read the *read.me* file. It contains the latest release notes, as well as information necessary to install the system.

Chapter 2

Type Definitions and Data Structures

SECTION 2.1 *Introduction*
2.2 *Short Integers*
2.3 *Long Integers*
2.4 *FIXPT and Floating Point Numbers*
2.5 *Character Strings*
2.6 *Screen Coordinates*
2.7 *2D Virtual Coordinates*
2.8 *3D Virtual Coordinates*
2.9 *Color Intensities*
2.10 *Graphics Cursor Drawing Attributes*
2.11 *Bit Maps*
2.12 *Font Specification*

2.1 Introduction

This chapter describes the data types and structures used by the LIB Shell.

For each data type and structure we give a brief description, the C declaration, and the equivalent form sent to the TMLS Interpreter and the LIB Shell. Note that the C-callable libraries and the TMLS Interpreter are **available only** with the PG-Series-1 and PG2-Series-1. If the parameter has a restricted range this is noted in the description.

We provide all users with the TMLS Interpreter. The TMLS Interpreter provides an easy way to communicate with the LIB Shell for testing and debugging purposes. You can also access the LIB Shell through one of several C-callable libraries. These libraries enable you to access the functionality of the LIB Shell from a C program.

> C Binding Only: You can not pass a structure to a function. A pointer to the structure is passed instead.

You read any return values from the Read FIFO. Refer to the Communications chapter in your product-specific supplemental manual for more details about communications.

2.2 Short Integers

A short is a signed 16-bit value. Many parameters use this data type, for example: angles, booleans, and flags. A negative short is in two's complement form. Values in the range -32768 to +32767 are possible.

The C binding defines the following types:

```
typedef short ANGLE;    /* angles outside the range
                           0-360 are reduced */
typedef short BOOL;     /* null=FALSE; non-null=TRUE */
```

An unsigned short is an unsigned 16-bit value. Values in the range 0 to 65535 are possible. Examples of this data type are color indices, masks, and transparency mode numbers.

The range of color indices depends on the number of bit planes in your system. In most cases, color indices are either 8-bit (0–255) or 4-bit (0–15) values.

2.3 Long Integers

A long is a signed 32-bit value, for example the size of command list memory blocks. An unsigned long is an unsigned 32-bit value. A line style is one example of this type. Parameters of this type are sent to the LIB Shell as two words: low word then high word.

2.4 FIXPT and Floating Point Numbers

A float is a value with an integer part and a fractional part in the range -32768.00000 to +32767.99999.

The C Binding defines the type:

```
typedef float FIXPT;    /* a 16.16 value */
```

A float is sent to the LIB Shell as two words.

- The first word contains the fractional part
- The second word contains the integer part

To derive the hexadecimal value from a given decimal value simply multiply the decimal value by 65536 and convert the result to hexadecimal. For example, 3.142 becomes:

$3.142 \times 65536 = 205914 =$ 0x0003 0x245A

where 0x0003 is the integer part and 0x245A is the fractional part. The value sent to the LIB Shell is 0x245A 0x0003.

This method is equally valid for calculating negative floats. Thus -3.142 becomes 0xFFFC 0xDBA6 and is sent as 0xDBA6 0xFFFC.

2.5 Character Strings

The C Binding and the TMLS Interpreter consider a string to be an array of type char. For example, a string containing the word *apple* is sent as `"apple"` . The LIB Shell expects a count, followed by ASCII character codes.

You send a character string to the LIB Shell as a list of character codes, preceded by a count parameter. The character codes to be sent depend on the current font.

- LIB Shell : count ch_code_1 ch_code_2 ch_code_3 . . . ch_code_{count}
- TMLS Interpreter : 'string' or "string"
- C binding : `char string[]`

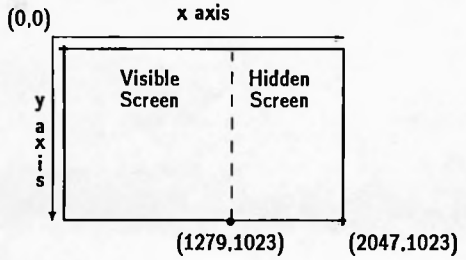
2.6 Screen Coordinates

This data type is used to store screen or physical coordinates. Screen memory is either visible or hidden.

- LIB Shell : x y (hex)
- TMLS Interpreter : x y (decimal)
- C Binding :

```
typedef struct phcoord {short x, y;} PHCOORD;
```

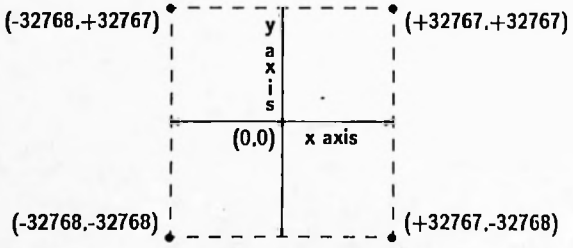
The default point of origin is the upper left corner of the screen. The diagram below illustrates the screen coordinate space for a system with 2 Kbyte × 1 Kbyte video RAM:



NOTE: The scsorg function lets you change the screen origin.

2.7 2D Virtual Coordinates

This data type is used to store 2D virtual coordinates. The default point of origin is the lower left corner of the screen:

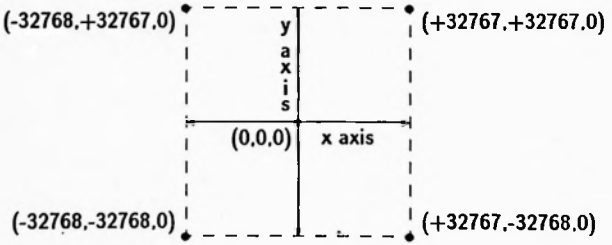


- LIB Shell : x_{fract} x_{int} y_{fract} y_{int} (hex)
- TMLS Interpreter : x y (decimal)
- C Binding :

```
typedef struct v2coord {FIXPT x, y;} V2COORD;
```

2.8 3D Virtual Coordinates

The V3COORD data structure stores 3D virtual coordinates. The origin point of the x, y, and z axes is the lower left corner of the screen. The positive z axis points to the viewer; the negative z axis points away from the viewer.



- LIB Shell : x_{fract} x_{int} y_{fract} y_{int} z_{fract} z_{int} (hex)
- TMLS Interpreter : x y z (decimal)
- C Binding :

```
typedef struct v3coord {FIXPT x, y, z;} V3COORD;
```

2.9 Color Intensities

A color is composed of three intensity values: red, green, and blue. Your system either supports 8-bit or 4-bit intensity values, depending on the type of DACs installed in your system. The former uses the 8 least significant bits of each intensity; the latter uses bits 4 to 7. Remaining bits are ignored.

- LIB Shell : red green blue (hex)
- TMLS Interpreter : red green blue (decimal)
- C Binding :

```
typedef struct /* the entry format in a Lookup Table */
      {
      unsigned short red;   /* red intensity   */
      unsigned short green; /* green intensity */
      unsigned short blue;  /* blue intensity  */
      } LUTVAL;
```

2.10 Graphics Cursor Drawing Attributes

The graphics cursor is drawn in two passes. This ensures that the cursor is always visible no matter what the screen contains. For instance a white cursor will disappear in a white circle. However, if you first draw it on a black background the cursor will remain visible.

LIB Shell : tdef1 mask1 fcol1 bcol1 bop1 tdef2 mask2 fcol2 bcol2 bop2 full_screen (hex)

TMLS Interpreter : tdef1 mask1 fcol1 bcol1 bop1 tdef2 mask2 fcol2 bcol2 bop2 full_screen (decimal)

C Binding :

```
typedef struct gcattr       /* cursor drawing attributes */
        {
        short tdef1;        /* first pass transparency
                               definition */
        short mask1;        /* first pass write
                               mask */
        short fcol1;        /* first pass foreground
                               color index */
        short bcol1;        /* first pass background
                               color index */
        short bop1;         /* first pass boolean
                               operator */
        short tdef2;        /* second pass transparency
                               definition */
        short mask2;        /* second pass write
                               mask */
        short fcol2;        /* second pass foreground
                               color index */
        short bcol2;        /* second pass background
                               color index */
        short bop2;         /* second pass boolean
                               operator */
        short full_screen   /* clipping flag */
        } GCATTR;
```

Five types of transparency are available:

- 0 = No transparency. Drawing is performed OPAQUE.
- 1 = Source-driven transparency. One bits in the source select pixels to be affected, while zero bits select pixels to be left unaffected by the drawing operation.
- 2 = Background Suppression. Same as OPAQUE mode except that pixels having the same value as the current background color are not drawn.
- 3 = Zero Suppression. If the color index resulting after the boolean operator and plane mask are applied to a pixel is zero, then the LIB Shell does not draw the pixel.
- 4 = Background-driven Transparency. A zero bit in the pattern signifies a pixel to be set to the current background color. However, pixels corresponding to one bits are unaffected.

The write mask selects which cursor bit planes are masked. A zero bit masks a plane i.e. the plane remains unchanged; a one bit allows you to write to the plane.

The following boolean operators are supported:

- 0 = Source → Destination(REPLACE)
- 1 = Source AND Destination → Destination(AND)
- 2 = Source AND !Destination → Destination
- 3 = All 0's → Destination(RESET)
- 4 = Source OR !Destination → Destination

- 5 = !(Source XOR Destination) → Destination(XNOR)
- 6 = !Destination → Destination(NOT Destination)
- 7 = !(Source OR Destination) → Destination(NOR)
- 8 = Source OR Destination → Destination(OR)
- 9 = Destination → Destination(NOOP)
- 10 = Source XOR Destination → Destination(XOR)
- 11 = !Source AND Destination → Destination
- 12 = All 1's → Destination(SET)
- 13 = !Source OR Destination → Destination
- 14 = !(Source AND Destination) → Destination(NAND)
- 15 = !Source → Destination(NOT Source)

The cursor foreground/background colors, transparency modes, and boolean operators, have the same range of values as the corresponding general drawing attributes. For more about general drawing attributes refer to Appendix B.

The clipping flag full_screen selects how the cursor is clipped. A null value disables full screen clipping i.e. the cursor is clipped with respect to the current clipping rectangle. A value of one means that the cursor is clipped with respect to the screen. A value of two means that the cursor is clipped with respect to the entire frame buffer.

2.11 Bit Maps

The BITMAP data structure is used to describe arbitrary raster objects. Examples of bit maps include the fill patterns, cursor shapes, raster text, and the frame buffer. Bit maps are rectangular. They can either be monochrome (1 bit/pixel) or polychrome (8 bits/pixel). The hardware must apply color expansion to monochrome bit maps before displaying them on the screen. It expands one bits into the current foreground color and zero bits into the current background color. The frame buffer is an example of a polychrome bit map.

For monochrome bit maps the pitch is the distance in words between two vertically contiguous elements. For polychrome bit maps set the pitch to 0; this informs the LIB Shell that the bit map is a polychrome one[1]. Bits 0 to 7 of the format element indicate the number of bits related to each pixel in each plane of the bit map. Bits 8 to 15 indicate the number of planes in the bit map.

The tl element is used to pass bit map data to the LIB Shell. If you are defining a monochrome bit map then data is passed in a buffer. If you are defining a polychrome bit map you must first draw it on the screen, then select that area of the screen as a polychrome bit map[2]. Refer to the lsdapat, lsdgc, and lsdfont command descriptions for the mechanics of defining fill patterns, cursor shapes, and raster fonts.

---

[1] The "Create Bit Map Storage Area" command, lscbm, is an exception to this rule.

[2] The "Fill Bit Map Storage Area" command, lsfbm, is an exception to the rule.

The LIB Shell has commands for creating, selecting, and removing bit map storage areas in Local RAM. Refer to the lscbm, lssbm, and lssrm command descriptions.

We support the following bit map types:

- **monochrome bit map:** The bit map below was defined as: width = 12 pixels; height = 4 scan lines; pitch = 1 word; format = 0x0101 (1 bit/pixel, 1 plane). Bit map data occupies 4 words.

BIT:	0															15
word 0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X
word 1	0	0	0	0	0	0	1	0	0	0	0	0	X	X	X	X
word 2	0	0	0	0	0	0	1	0	0	0	0	0	X	X	X	X
word 3	1	1	1	1	1	1	1	1	1	1	1	1	X	X	X	X

Bit Map Format (X=don't care)

- **polychrome bit map:** This bit map was defined as: width = 6 pixels; height = 2 scan lines; pitch = 0; format = 0x0108 (8 bits/pixel, 1 plane). Bit map data occupies 6 words.

byte 0			. . .		byte 5
P_0	P_1	P_2	P_3	P_4	P_5
P_6	P_7	P_8	P_9	P_{10}	P_{11}
byte 6			. . .		byte 11

Extra planes can be used to store a graphics overlay or the graphics cursor.

LIB Shell : w h ptch fmt data (hex) where data is a buffer containing the bit map data or w h ptch fmt x y (hex) where (x,y) is a screen coordinate.

TMLS Interpreter : w h ptch fmt data (decimal) where data is a buffer containing the bit map data or w h ptch fmt x y (decimal) where (x,y) is a screen coordinate.

C Binding :

```
typedef struct  {/* arbitrary bit map descriptor */
      unsigned short w;     /* width in pixels            */
      unsigned short h;     /* height in pixels           */
      unsigned short ptch; /* pitch in words              */
      unsigned short fmt;  /* format:
                              8 LSBs: # of bits/pixel
                              8 MSBs: # of planes        */
      short *tl;           /* pointer to top-left
                              corner of data buffer or
                              to an (x,y) screen coord. */
      } BITMAP;
```

2.12 Font Specification

A font definition is stored in the form of a FONT structure.

LIB Shell : fmt firstch lastch_off replch_off brkch_off width height pitch nkpairs ascent half base descent offset$_0$ width$_0$ . . . offset$_n$ width$_n$ firstch$_1$ nextch$_1$ amount$_1$ . . . firstch$_{nkpairs}$ nextch$_{nkpairs}$ amount$_{nkpairs}$ data (hex)

TMLS Interpreter : fmt firstch lastch_off replch_off brkch_off width height pitch nkpairs ascent half base descent offset$_0$ width$_0$ . . . offset$_n$ width$_n$ firstch$_1$ nextch$_1$ amount$_1$ . . . firstch$_{nkpairs}$ nextch$_{nkpairs}$ amount$_{nkpairs}$ data (decimal)

C Binding :

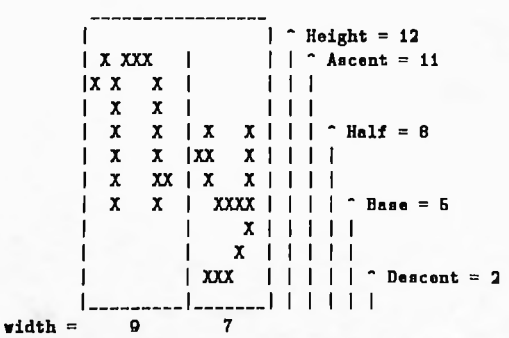
```
typedef struct font /* font */
        {
        short fmt;          /* character format:
                               1 = raster character.
                               2 = small stroke font.
                               4 = large stroke font.        */
        short firstch;      /* first character code.         */
        short lastch_off;   /* last character as offset
                               from first character.         */
        short replch_off;   /* replacement char as offset
                               from the first character.     */
        short brkch_off;    /* word delimiter as offset
                               from the first character.     */
        short width;        /* width of characters.
                               Null for proportional-spaced. */
        short height;       /* character height              */
        short pitch;        /* font pitch in words.          */
        short nkpairs;      /* number of kerned pairs.       */
        short ascent;       /* ascent distance (from
                               bottom) in bits               */
        short half;         /* half distance (from bottom)
                               in bits                       */
        short base;         /* base distance (from bottom) in
                               bits                          */
        short descent;      /* descent (from bottom) in bits */
        short *owtable;     /* offset/width table address.
                               Null for mono-spaced.         */
        short *kerntable;   /* pointer to kerning table      */
        short *data;        /* font data block address       */
        } FONT;
```

Please note the following points:

- The value of *n* is the number of characters in the font.
- The offset width table is not required for mono-spaced raster fonts. Some types of fonts only require an offset table. Refer to the table in Subsection 2.12.1.
- data is a pointer to a buffer for the C binding. For TMLS and the LIB Shell it is the actual buffer data.

Characters in a font are numbered consecutively, starting with the character code stored in the firstch element. The replacement character is used if you specifiy an undefined character code. The LIB Shell interprets the break character to mean it has encountered the end of a word. The break character is the basis of the LIB Shell's word placement algorithms.

The diagram below illustrates the dimensions of two proportional spaced characters:



The data entry is a buffer (or a pointer to the buffer) containing the definitions of the characters in the font. For a raster font, the buffer is a monochrome bit map. The first bit map row defines the top row of all characters in the font, and so on down to the last row, which defines the bottom row of all characters. The height of the bit map is the height of a character cell and the width of the bit map is the sum of the widths of all the characters in the font. Thus, each character row is apart by the width of the raster font bit map in bits (i.e. the font pitch). If the width of the bit map in bits in not a multiple of 16 then you must add enough bits after the last character definition to make it so.

For a stroke font, the buffer contains a series of move pairs and pen-up values. A stroke character is defined as a series of relative move pairs (Δx_i, Δy_i), each one making a stroke of the character. A move is a signed 8-bit value for small-definition fonts, and a signed 16-bit value for large definition fonts. Thus, each move pair occupies 16 bits or 32 bits. A special pen-up value (-128 for small fonts, -32768 for large fonts) is required to signal that stroke drawing should resume at the next move pair. The pitch of a stroke font is the number of words in the font data block.

A font can be either mono-spaced, or proportionally-spaced. Proportional spacing is signalled by a null width in the width element. This brings the total number of font types to six.
i.e. raster - mono/proportional spaced
small stroke - mono/proportional spaced
large stroke - mono/proportional spaced

2.12.1 Text Offset Table Entry

The LIB Shell uses the offset/width table to determine where each character definition starts in the font data buffer. The offset is the distance from the start of the font data block to the desired character definition. The width is the width of the character.

The type of the font determines what kind of table is required:

Font:	Offset:	Width:
mono-spaced raster	Not required.	Not required.
proportional-spaced raster	Offset in bits from the start of font data block.	Not required.
small definition mono-spaced stroke	Offset in bytes to start of definition.	Not required.
large definition mono-spaced stroke	Offset in words to start of definition.	Not required.
small definition proportional-spaced stroke	Offset in bytes to start of definition.	Width of character in pixels.
large definition proportional-spaced stroke	Offset in words to start of definition.	Width of character in pixels.

This table has an entry for each character in the font, plus one. Set the last offset entry to the number of bytes in the font data block.

LIB Shell : $offset_i$ $width_i$ (hex)

TMLS Interpreter : offset; width; (decimal)

C Binding :

```
typedef struct /* offset/width table entry */
        {
        short offset;           /* offset in font */
        short width;            /* character width */
        } OWENTRY;
```

2.12.2 Kerned Pair Entry

This data structure defines an entry in a kerning table. The FONT.nkpairs element contains the number of kerned pairs in the font. The amount entry is the number of pixels of overlap between the first and second character in the kerned pair.

LIB Shell : firstch; nextch; amount; (hex)

TMLS Interpreter : firstch; nextch; amount; (decimal)

C Binding :

```
typedef struct /* kerned pair entry */
        {
        short firstch;          /* first character code
                                   of kerned pair */
        short nextch;           /* last character code
                                   of kerned pair */
        short amount;           /* kerning amount in pixels
                                   for this pair */
        } KPAIR;
```

Chapter 3

Command Descriptions

This chapter contains the command descriptions for the LIB Shell, C Binding, and the TMLS Interpreter.

Chapter 2 explains the data types and structures that the commands use. Appendix A lists default parameter values, Appendix B summarizes the drawing attributes, Appendix C provides a summary of LIB Shell Commands, and Appendix D lists the command opcodes. Please note that there are some LIB Shell commands which are specific to a particular board set (for example, the PG-Series-1 lsegasw command). Refer to your supplemental manual for details.

We give a sample command description here, followed by the LIB Shell command descriptions in alphabetical order. Commands marked with the icon

can be used in command lists and are stored in the command list definition, whereas ones marked with

can be used in command lists, but are not stored in the command list definition. Instead, they are executed as the command list is built. Finally, this icon

indicates commands that can never be used in command lists. These commands will be ignored if sent during command list building.

SAMPLE Sample Command Description

Format

LIB Shell: A 16-bit hexadecimal opcode, followed by the names of the parameters. The LIB Shell only accepts 16-bit hexadecimal values.

C Binding: The C function declaration with the formal parameters. Available only for the PG-Series-1, PG2-Series-1, and SM-Series-1.

TMLS Interpreter: The TMLS Interpreter command and parameter names. The Interpreter operates in ASCII. Available only for the PG-Series-1, PG2-Series-1, and SM-Series-1.

Parameters Each parameter is described here; what its purpose is and what values are expected.

Description What the command does and when to use it. The icon at the top of the page means this command should not be used in a command list.

Note Any special command requirements would be mentioned here.

Return Value Describes what the command returns, if anything.

See 'Also Names of commands which perform similar functions or which influence the result of this command.

KRGCDI Disable Graphics Cursor

Format

- LIB Shell:0x0084
- C Binding:void krgcdi();
- TMLS Interpreter: krgcdi

Parameters None.

Description

This function disables the cursor. It automatically removes it from the screen if it was drawn.

Return Value None.

Error None.

See Also krgcen, krgcmv

Enable Graphics Cursor KRGCEN

Format

- LIB Shell: 0x0083
- C Binding: void krgcen();
- TMLS Interpreter: krgcen

Parameters None.

Description

This function enables the cursor. It automatically draws the cursor at the current cursor position. The default cursor position is (0,0).

Note If a user-defined cursor has not been defined and selected then the default cursor (label = -1) is used.

The cursor should be either disabled or frozen during video memory window accesses.

Return Value None.

Error None.

See Also lsdgc, lssgc, krgcdi, krgcmv, krgcfr

KRGCFR Freeze Cursor

Format

- LIB Shell: 0x007E
- C Binding: void krgcfr();
- TMLS Interpreter: krgcfr

Parameters None.

Description

This command disables the cursor but leaves it displayed on the screen. It freezes the cursor. Subsequent drawings in the region of the cursor will overwrite the cursor.

Note It is necessary to either disable or freeze the cursor during video memory window accesses. The krgcfr command is useful if the window access is occurring outside the cursor region and you do not want the cursor to disappear during the access.

Return Value None.

Error None.

See Also krgcdi

Move Graphics Cursor KRGCMV

Format

- LIB Shell: 0x0085 x y
- C Binding: void krgcmv(x, y);
- TMLS Interpreter: krgcmv x, y

Parameters

- short *x:* (Default = 0) the x-coordinate of the new cursor position.
- short *y:* (Default = 0) the y-coordinate of the new cursor position.

Description

This function moves the graphics cursor so that its hot-spot lies over the specified position. It draws the cursor according to the current graphics cursor shape and attributes. If the cursor is currently disabled, the new cursor position is remembered but no drawing occurs.

Return Value None.

Error None.

See Also krgcen, krgcdi

KRGCSDA Graphics Cursor Attributes

Format

- LIB Shell: 0x0082 gcspec
- C Binding: void krgcsda(gcspecptr);
- TMLS Interpreter: krgcsda gcspec

Parameters

- GCATTR gcspec: (see Appendix A for default value) the graphics cursor drawing attributes.
- GCATTR *gcspecptr: a pointer to the GCATTR structure containing the graphics cursor specification. Refer to Section 2.10 for a description of this structure.
- HGCATTR gcspec: (cursor display mode = 2 only) the graphics cursor drawing attributes used by the cursor controller chip of boards using a harware cursor. See your supplemental manual for more information.
- HGCATTR *gcspecptr: (cursor display mode = 2 only) a pointer to the graphics cursor drawing attributes used by the cursor controller chip of those boards using a hardware cursor. See your supplemental manual for more information.

Description

This function provides control over how the LIB Shell draws the cursor.

Return Value None.

Error None.

See Also lsdgc, krgcsdm

Set Cursor Display Mode KRGCSDM

Format

- LIB Shell: 0x0086 disp_mode
- C Binding: void krgcsdm(disp_mode);
- TMLS Interpreter: krgcsdm disp_mode

Parameters

short disp_mode: (Default = 0) the display mode:

- 0: Indicates that the cursor is drawn in each frame. The cursor can be used without reserved planes.
- 1: Indicates that the cursor is redrawn only when it moves. The cursor must have reserved planes.
- 2: Indicates that cursor drawing is handled by the cursor controller chip (for boards with a hardware cursor only).
- 3: Indicates that the cursor is to be drawn in the overlay, at each frame (for boards with dedicated overlay only).

Description

This command sets the cursor display mode according to the parameter description above.

Note

You must select the display mode and the display attributes before enabling the cursor.

KRGCSDM Set Cursor Display Mode

Return Value None.

Error None.

See Also krgcsda

Clear Clipping Rectangle KRHWCCLR

Format

- LIB Shell: 0x0023 color_index
- C Binding: void krhwcclr(color_index);
- TMLS Interpreter: krhwcclr color_index

Parameters

- unsigned short *color_index:* the color index to use for the clear operation.

Description

This function clears the clipping rectangle interior to a given color.

Return Value None.

Error None.

See Also krhwfclr, krhwsclr, krhwrclr

KRHWCPRSS Raster Copy with Rotation

Format

- LIB Shell: 0x002F spx spy dpx dpy sizex sizey theta
- C Binding: void krhwcprss(spx, spy, dpx, dpy, sizex, sizey theta);
- TMLS Interpreter: krhwcprss spx spy dpx dpy sizex sizey theta

Parameters

- short spx: the x-coordinate of the top left corner of the source area to be copied.
- short spy: the y-coordinate of the top left corner of the source area to be copied.
- short dpx: the x-coordinate to which *spx* is mapped on the screen.
- short dpy: the y-coordinate to which *spy* is mapped on the screen.
- short sizex: the width of the area to be copied. A negative value will cause this command to be ignored.
- short sizey: the height of the area to be copied. A negative value will cause this command to be ignored.
- ANGLE theta: the rotation angle by which the original raster is rotated about the destination point. Angles are reduced to the nearest 90° increment: e.g. -189° → -180°

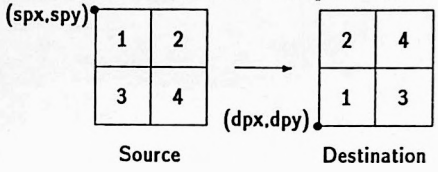
Description

This function copies a rectangular area of the screen to another part on the screen. At the same time, it rotates the contents of the bit map by the specified number of degrees.

Raster Copy with Rotation KRHWCPRSS

The result of this command is equivalent to copying the source raster to the destination point and rotating it about this point by angle *theta.*

The example below shows a bit map rotated 90°:



Overlap tests and adjustments are performed.

Return Value None.

Error None.

See Also krhwcpss, krhwcpxmss, krhwcpymss

KRHWCPSS Copy a Bitmap (Screen-Screen)

Format

- LIB Shell: 0x0027 spx spy dpx dpy sizex sizey
- C Binding: void krhwcpss(spx, spy, dpx, dpy, sizex, sizey);
- TMLS Interpreter: krhwcpss spx spy dpx dpy sizex sizey

Parameters

- short spx: the x-coordinate of the top left corner of the source area to be copied.
- short spy: the y-coordinate of the top left corner of the source area to be copied.
- short dpx: the x-coordinate of the top left corner of the destination area on the screen.
- short dpy: the y-coordinate of the top left corner of the destination area on the screen.
- short sizex: the width of the area to be copied. A negative value will cause this command to be ignored.
- short sizey: the height of the area to be copied. A negative value will cause this command to be ignored.

Description

This function copies a rectangular area of the screen to another part on the screen. Overlap tests and adjustments are performed.

Return Value None.

Error None.

See Also krhwcprss, krhwcpxmss, krhwcpymss

Mirror Image Over X — KRHWCPXMSS

Format

- LIB Shell: 0x0030 spx spy dpx dpy sizex sizey
- C Binding: void krhwcpxmss(spx, spy, dpx, dpy, sizex, sizey);
- TMLS Interpreter: krhwcpxmss spx spy dpx dpy sizex sizey

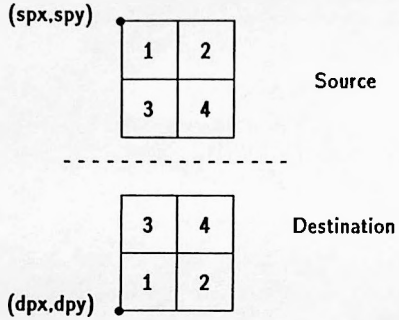
Parameters

- short spx: the x-coordinate of the top left corner of the source area to be copied.
- short spy: the y-coordinate of the top left corner of the source area to be copied.
- short dpx: the x-coordinate of the bottom left corner of the destination area on the screen.
- short dpy: the y-coordinate of the bottom left corner of the destination area on the screen.
- short sizex: the width of the area to be copied. A negative value will cause this command to be ignored.
- short sizey: the height of the area to be copied. A negative value will cause this command to be ignored.

Description

This function copies a rectangular area of the screen to another part of the screen. At the same time, it performs a mirror transformation of the contents of the rectangle with respect to the x-axis:

KRHWCPXMSS Mirror Image Over X



Overlap tests and adjustments are performed.

Return Value None.

Error None.

See Also krhwcprss, krhwcpss, krhwcpymss

Mirror Image Over Y KRHWCPYMSS

Format

- LIB Shell: 0x0031 spx spy dpx dpy sizex sizey
- C Binding: void krhwcpymss(spx, spy, dpx, dpy, sizex, sizey);
- TMLS Interpreter: krhwcpymss spx spy dpx dpy sizex sizey

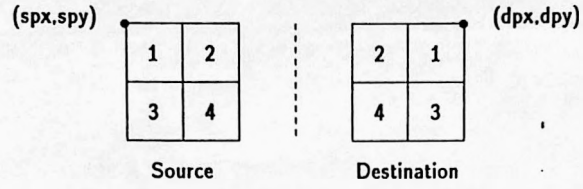
Parameters

- short spx: the x-coordinate of the top left corner of the source area to be copied.
- short spy: the y-coordinate of the top left corner of the source area to be copied.
- short dpx: the x-coordinate of the top right corner of the destination area on the screen.
- short dpy: the y-coordinate of the top right corner of the destination area on the screen.
- short sizex: the width of the area to be copied. A negative value will cause this command to be ignored.
- short sizey: the height of the area to be copied. A negative value will cause this command to be ignored.

Description

This function copies a rectangular area of the screen to another part on the screen. At the same time, it performs a mirror transformation of the contents of the rectangle with respect to the y-axis:

KRHWCPYMSS Mirror Image Over Y



Overlap tests and adjustments are performed.

Return Value None.

Error None.

See Also krhwcprss, krhwcpss, krhwcpxmss

Clear Frame Buffer — KRHWFCLR

Format

- LIB Shell: 0x0021 color_index
- C Binding: void krhwfclr(color_index);
- TMLS Interpreter: krhwfclr color_index

Parameters

- unsigned short color_index: the frame buffer is cleared to this color index.

Description

This function clears a 2K × 1K frame buffer to the given color index. The OPAQUE transparency mode and the REPLACE boolean operator are used. The current plane mask is ignored: all planes are affected by the operation.

Note For boards with 2K × 2K frame buffers, you must first select which half of the frame buffer to clear by using the krhwsfcm command.

Return Value None.

Error None.

See Also krhwcclr, krhwrclr, krhwsfcm, krhwsclr

KRHWGCLS Get Current Line Style

Format

- LIB Shell: 0x0014
- C Binding: void krhwgcls();
- TMLS Interpreter: krhwgcls

Parameters None.

Description

This function reports the current 32-bit line style.

Return Value The current line style is returned in the Read FIFO as a hexadecimal value. The host reads the data as follows:

- LIB Shell: Read data from the Read FIFO entrance.
- C Binding: Use the iogetw function of the C Binding to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also krhwgcls, krhwsils, krhwrcls

Get Pixel KRHWGPIX

Format

- LIB Shell: 0x0028 x y
- C Binding: void krhwgpix(x, y);
- TMLS Interpreter: krhwgpix x, y

Parameters

- short x: the x-coordinate of the pixel.
- short y: the y-coordinate of the pixel.

Description

This function returns the pixel value under the given frame buffer position. The clipping rectangle is ignored, the plane mask is effective, and the specified position is not validated.

Return Value The pixel's value is returned in the Read FIFO a hexadecimal value. The valid width of the value depends on the number of bit planes in your system; common widths are 4 or 8 bits. Other bits are left undefined. The host reads the data as follows:

- LIB Shell: Read data from the Read FIFO entrance.
- C Binding: Use the iogetw function of the C Binding to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also krhwspix

KRHWRCLR Clear Rectangle

Format

- LIB Shell: 0x0024 left top right bottom color_index
- C Binding: void krhwrclr(left, top, right, bottom, color_index);
- TMLS Interpreter: krhwrclr left top right bottom color_index

Parameters

- short left: the x-coordinate of the top left corner of the rectangle.
- short top: the y-coordinate of the top left corner of the rectangle.
- short right: the x-coordinate of the bottom right corner of the rectangle.
- short bottom: the y-coordinate of the bottom right corner of the rectangle.
- unsigned short color_index: the color index to use for clear operation.

Description

This function clears a screen rectangle to the specified color index.

Return Value None.

Error None.

See Also krhwfclr, krhwsclr, krhwcclr

Reset Current Line Style KRHWRCLS

Format

- LIB Shell: 0x0016
- C Binding: void krhwrcls();
- TMLS Interpreter: krhwrcls

Parameters None.

Description

This function resets the current line style to the initial line style.

Return Value None.

Error None.

See Also krhwsils, krhwgcls, krhwscls

KRHWSBCOL Set Background Color

Format

- LIB Shell: 0x001A color_index
- C Binding: void krhwsbcol(color_index);
- TMLS Interpreter: krhwsbcol color_index

Parameters

- unsigned short color_index: (Default = 0) the background color index.

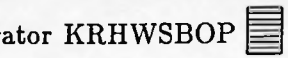
Description

This function sets the background color to the specified index.

Return Value None.

Error None.

See Also krhwsbop, krhwsclipr, krhwsfcol, krhwsmsk, krhwstdef

Set Graphics Boolean Operator KRHWSBOP 

Format

- LIB Shell: 0x001B bool_operator
- C Binding: void krhwsbop(bool_operator);
- TMLS Interpreter: krhwsbop bool_operator

Parameters

- short bool_operator: (Default = 0) the boolean operator code; the following operators are supported:
  - 0: Source → Destination (REPLACE)
  - 1: Source AND Destination → Destination (AND)
  - 2: Source AND !Destination → Destination
  - 3: All 0's → Destination (RESET)
  - 4: Source OR !Destination → Destination
  - 5: !(Source XOR Destination) → Destination (XNOR)
  - 6: !Destination → Destination (NOT Destination)
  - 7: !(Source OR Destination) → Destination (NOR)
  - 8: Source OR Destination → Destination (OR)
  - 9: Destination → Destination (NOOP)
  - 10: Source XOR Destination → Destination (XOR)
  - 11: !Source AND Destination → Destination
  - 12: All 1's → Destination (SET)
  - 13: !Source OR Destination → Destination
  - 14: !(Source AND Destination) → Destination (NAND)
  - 15: !Source → Destination (NOT Source)

KRHWSBOP Set Graphics Boolean Operator

Description

This function sets the graphics boolean operator.

Return Value None.

Error None.

See Also krhwsbcol, krhwsclipr, krhwsfcol, krhwsmsk, krhwstdef

Set Clipping Rectangle KRHWSCLIPR

Format

- LIB Shell: 0x001E left top right bottom
- C Binding: void krhwsclipr(left, top, right, bottom);
- TMLS Interpreter: krhwsclipr left top right bottom

Parameters

- short left: (Default = 0) the x-coordinate of the top left corner of the rectangle.
- short top: (Default = 0) the y-coordinate of the top left corner of the rectangle.
- short right: (Default = 1279, 1023, or 639) the x-coordinate of the bottom right corner of the rectangle.
- short bottom: (Default = 1023, 767, or 479) the y-coordinate of the bottom right corner of the rectangle.

Description

This function specifies the inclusive boundaries of the clipping rectangle. The specified values become the new inclusive hardware clipping boundaries. All subsequent drawing operations are guaranteed not to extend outside it. Rectangle corner checking is automatically performed, using the width and height of the frame buffer as references.

Return Value None.

Error None.

See Also lsdclpl, krhwsbcol, krhwsbop, krhwsfcol, krhwsmsk, krhwstdef

KRHWSCLR Clear Visible Screen

Format

- LIB Shell: 0x0022 color_index
- C Binding: void krhwsclr(color_index);
- TMLS Interpreter: krhwsclr color_index

Parameters

- unsigned short color_index: the visible screen is cleared to this color index.

Description

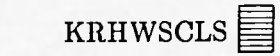
This function clears the entire visible screen to the specified color index.

Return Value None.

Error None.

See Also krhwfclr, krhwcclr, krhwrclr

Set Current Line Style — KRHWSCLS



Format

- LIB Shell: 0x0015 line_style
- C Binding: void krhwscls(line_style);
- TMLS Interpreter: krhwscls line_style

Parameters

- unsigned long line_style: a 32-bit line style value.

Description

This function sets the current line style. The next series of outline primitives you draw will use this line style. Subsequently, the line style reverts to the initial line style set by the last krhwsils command.

Note Setting the current line style will only be reflected if line style continuity has been enabled using scolsmode. Otherwise, the line style set by krhwsils will be used.

Return Value None.

Error None.

See Also krhwsils, krhwrcls, krhwgcls, scolsmode

KRHWSDIRECT Set Window Access Mode

Format

- LIB Shell: 0x0017 mode
- C Binding: void krhwsdirect(mode);
- TMLS Interpreter: krhwsdirect mode

Parameters

- short mode: the access mode for the video memory windows. The present system only supports those access modes which use planes 0 to 7; planes 8 to 15 are not available. The plane mask must be set appropriately. Only enable those planes which are currently accessable. The following table lists possible values for mode. Bits 7 to 15 are not used.

Bits 6-0	bits per pixel	Access to plane(s)	Bits 6-0	bits per pixel	Access to plane(s)
000 1111	16	0 to 15	011 0000	1	0
001 0000	8	0 to 7	011 0001	1	1
001 1000	8	8 to 15	011 0010	1	2
010 0000	4	0 to 3	011 0011	1	3
010 0100	4	3 to 7	011 0100	1	4
010 1000	4	8 to 11	011 0101	1	5
010 1100	4	12 to 15	011 0110	1	6
110 0000	2	0 to 1	011 0111	1	7
110 0010	2	2 to 3	011 1000	1	8
110 0100	2	4 to 5	011 1001	1	9
110 0110	2	6 to 7	011 1010	1	10
110 1000	2	8 to 9	011 1011	1	11
110 1010	2	10 to 11	011 1100	1	12
110 1100	2	12 to 13	011 1101	1	13
110 1110	2	14 to 15	011 1110	1	14
–	–	–	011 1111	1	15

Set Window Access Mode KRHWSDIRECT

Description

This function sets the access mode for the video memory windows. For example, to access the 4 Kbyte window:

1. Use krhwsmsk to set the appropriate plane mask.
2. Use krhwsdirect to select the desired access mode. The plane access mode selected here must reflect the plane mask selected in step 1.
3. Send an lsnoop command immediately after and wait for the Write FIFO to go empty.
4. Use the PAGE register to select which scan line appears in the window.
5. (PG-Series-1 only) Enable the 4 Kbyte window by writing a zero to bit 6 of the MODE register.
6. Send data to the window.

For more details refer to the Communications chapter in the supplemental manual for your board.

Note The access mode may change after the next command is transmitted to the Write FIFO.

Return Value None.

Error None.

See Also krhwsdirect, krhwsmsk, lsnoop

KRHWSFCM Set Flash Clear Mode

Format

- LIB Shell: 0x0010
- C Binding: void krhwsfcm(mode);
- TMLS Interpreter: krhwsfcm mode

Parameters

- unsigned short mode: the flash clear mode (default = 0).

low byte: Planes selection option.

bit 0:

0 = Enable write on planes 0-3 during flash clear.

1 = Disable write on planes 0-3 during flash clear.

bit 1:

0 = Enable write on planes 4-7 during flash clear.

1 = Disable write on planes 4-7 during flash clear.

high byte: Frame buffer select option.

null = top 2Kx1K portion of the frame buffer.

non-null = bottom 2Kx1K portion of the frame buffer.

The non-null option is only meaningful for products which have a 2Kx2K frame buffer. When it is non-null, the flash clear will operate on the bottom 2Kx1K portion of the frame buffer.

Set Flash Clear Mode KRHWSFCM

Description

This function sets the flash clear mode. The flash clear mode determines what portion of the frame buffer is cleared by the krhwfclr command.

Return Value None.

Error None.

See Also krhwfclr

KRHWSFCOL Set Foreground Color

Format

- LIB Shell: 0x0019 color_index
- C Binding: void krhwsfcol(color_index);
- TMLS Interpreter: krhwsfcol color_index

Parameters

- unsigned short color_index: (Default = 255) the foreground color index.

Description

This function sets the foreground color to the specified index.

Return Value None.

Error None.

See Also krhwsbcol, krhwsbop, krhwsclipr, krhwsmsk, krhwstdef

Set Initial Line Style — KRHWSILS

Format

- LIB Shell: 0x0038 new_style
- C Binding: void krhwsils(new_style);
- TMLS Interpreter: krhwsils new_style

Parameters

- unsigned long new_style: (Default = 0xFFFF 0xFFFF) a 32-bit line style value.

Description

This function sets the initial outlining style to the supplied value. At a given point in time this may or may not be the current line style. You can reset the current line style to the initial line style by issuing a krhwrcls command.

Use the scolsmode command to enable or disable line style continuity. If you draw a series of outlines with line pattern continuity enabled then the line pattern will appear continous. Otherwise, the shell resets the current line style to the initial line style each time you draw an outline primitive.

Return Value None.

Error None.

See Also krhwrcls, krhwscls, krhwgcls, scolsmode

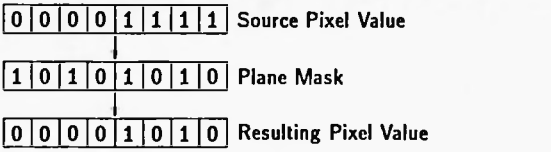
KRHWSMSK — Set Plane Mask

Format

- LIB Shell: 0x001C mask
- C Binding: void krhwsmsk(mask);
- TMLS Interpreter: krhwsmsk mask

Parameters

- unsigned short mask: (Default = 0xFFFF) a pixel written to or read from the frame buffer must pass through the plane mask. When you read a pixel the system ANDs it with the plane mask:



Writing a pixel is more complicated. First the pixel is affected by the current boolean operator and transparency mode. The resulting pixel value is compared to the plane mask: a zero bit in the i^{th} position masks the i^{th} plane (do not write the bit), while a one bit means to write the bit. You should refer to the appropriate section in the *The LIB Shell Programmer's Manual* for more details.

Set Plane Mask — KRHWSMSK

Description

This function sets the plane mask.

Return Value None.

Error None.

See Also krhwsbcol, krhwsbop, krhwsclipr, krhwsfcol, krhwstdef

KRHWSPIX — Set a Pixel

Format

- LIB Shell: 0x0029 x y color_index
- C Binding: void krhwspix(x, y, color_index)
- TMLS Interpreter: krhwspix x y color_index

Parameters

- short x: the x-coordinate of the pixel.
- short y: the y-coordinate of the pixel.
- unsigned short color_index: the color index to set the pixel.

Description

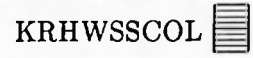
This function sets the pixel value under the given frame buffer position to the specified color index.

Return Value None.

Error None.

See Also krhwgpix

Set Seek Color — KRHWSSCOL



Format

- LIB Shell: 0x002E color_index
- C Binding: void krhwsscol(color_index);
- TMLS Interpreter: krhwsscol color_index

Parameters

- unsigned short color_index: (Default = 0) the color index to search for in pixel seek operations.

Description

This function sets the pixel-seek color. The pixel data read from the frame buffer is ANDed with the current plane mask (set by the krhwsmsk command) and the current seek mask (set by the krhwssmsk command) before the LIB Shell compares it to the pixel-seek color.

Return Value None.

Error None.

See Also krhwsmsk, krhwwfps, krhwwbps, krhwufps, krhwubps, krhwssmsk

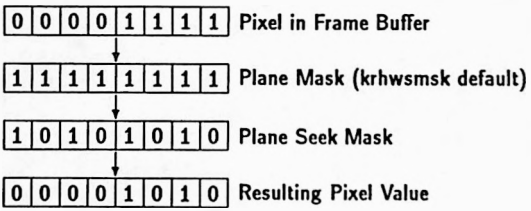
KRHWSSMSK Set Graphics Seek Mask

Format

- LIB Shell: 0x001D mask
- C Binding: void krhwssmsk(mask);
- TMLS Interpreter: krhwssmsk mask

Parameters

- unsigned short mask: (Default = 0xFFFF) the seek mask. During pixel seek operations the shell ANDs each pixel with this mask before comparing it to the seek color. The example below shows a pixel being read from the frame buffer:


0 0 0 0 1 1 1 1
Pixel in Frame Buffer
1 1 1 1 1 1 1 1
Plane Mask (krhwsmsk default)
1 0 1 0 1 0 1 0
Plane Seek Mask
0 0 0 0 1 0 1 0
Resulting Pixel Value

Set Graphics Seek Mask KRHWSSMSK

Description

This function sets the plane mask used for pixel-seek operations. The pixel read from the frame buffer is first ANDed with the current plane mask and then the plane seek mask to obtain the resulting pixel value. The resulting pixel value is then compared with the seek color set by krhwsscol.

Return Value None.

Error None.

See Also krhwsscol, krhwwfps, krhwwbps, krhwufps, krhwubps

KRHWSTDEF Set Transparency Mode

Format

- LIB Shell: 0x0018 mode
- C Binding: void krhwstdef(mode);
- TMLS Interpreter: krhwstdef mode

Parameters

- unsigned short mode: (Default = 0) the transparency mode. The following values are expected:
  - 0: No transparency (OPAQUE). Pixels corresponding to one bits in the pattern are set to the current foreground color. Pixels corresponding to zero bits are set to the current background color.
  - 1: Source-driven Transparency. A one bit in the pattern signifies a pixel to be set to the current foreground color. However, pixels corresponding to zero bits are unaffected.
  - 2: Background Suppression. Same as OPAQUE mode except that pixels having the same value as the current background color are not drawn.
  - 3: Zero Suppression. If the color index resulting after the boolean operator and plane mask are applied to a pixel is zero, then the shell does not draw the pixel.
  - 4: Background-driven Transparency. A zero bit in the pattern signifies a pixel to be set to the current background color. However, pixels corresponding to one bits are unaffected.

Set Transparency Mode KRHWSTDEF

Description

This function sets the transparency operation to behave in one of several ways. Transparency is only applied to the following patterns: line patterns, raster text, monochrome wide line fill patterns, and monochrome area fill patterns. It also affects the following raster operations: krhwspix, krhwcprss, krhwcpss, krhwcpxmss, krhwcpymss, and lscpbmss.

Source-driven and background-driven transparency are particulary useful for drawing fill patterns, line patterns, and graphics cursors.

Notes

- The transparency mode is ignored for polychrome fill patterns.
- The graphics cursor's transparency modes are set independently, refer to the lsdgc command description.

Return Value None.

Error None.

See Also krhwsbcol, krhwsbop, krhwsclipr, krhwsfcol, krhwsmsk, lsdgc

KRHWSZP — Set Zoom Pan

Format

- LIB Shell: 0x0011 zoomx zoomy panx pany finepanx finepany
- C Binding: void krhwszp(zoomx, zoomy, panx, pany, finepanx, finepany);
- TMLS Interpreter: krhwszp zoomx zoomy panx pany finepanx finepany

Parameters

- short zoomx: the zooming factor along the x-axis.
- short zoomy: the zooming factor along the y-axis.
- short panx: the x-coordinate of the upper left corner of the pan area in screen coordinates.
- short pany: the y-coordinate of the upper left corner of the pan area in screen coordinates.
- short finepanx: the relative x-coordinate of the upper left corner of the fine pan zoomed pixel area.
- short finepany: the relative y-coordinate of the upper left corner of the fine pan zoomed pixel area.

Description

This command allows you to simultaneously zoom and pan the contents of the frame buffer. The zoomx and zoomy parameters allow you to control the zoom factors in the horizontal and vertical directions respectively. The coordinate (panx,pany) becomes the new display origin (top left corner of the screen).

The finepanx and finepany parameters serve as relative offsets when panning a zoomed pixel area.

Set Zoom Pan — KRHWSZP

The parameters of this command are absolute, meaning they are not relative to the previous zoom/pan operation.

	Accepted Values	
	Without ZPGA	With ZPGA
zoomx	1	1, 2, 4, 8, 16
zoomy	1, 2, 4, 8	1, 2, 4, 8
panx	any multiple of 8 x-coordinate	any x-coordinate
pany	any y-coordinate	any y-coordinate
finepanx	0	between 0 and *zoomx* - 1
finepany	0	0

Return Value None.

Error None.

See Also

KRHWUBPS Until-Backwards Pixel Seek

Format

- LIB Shell: 0x002D sy xstart xend
- C Binding: void krhwubps(sy, xstart, xend);
- TMLS Interpreter: krhwubps sy xstart xend

Parameters

- short sy: the vertical coordinate of the line to be searched.
- short xstart: the horizontal start coordinate for the search.
- short xend: the horizontal limit coordinate for the search.

Description

This function is used to implement a pixel-seek operation. It seeks a preset seek color along a horizontal screen line, seeking to the left (backwards). The LIB Shell ANDs each pixel with the current seek mask before comparing it to the seek color. The seek continues until either the seek color is found or *xend* is reached without finding the seek color.

Until-Backwards Pixel Seek KRHWUBPS

Return Value

The horizontal coordinate of the first pixel satisfying the search condition along the backwards search direction. If the search limit was reached before the search terminated, the value returned is one pixel past the limit. The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the iogetw function of the C Binding to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also krhwwfps, krhwwbps, krhwufps, krhwssmsk, krhwsscol

KRHWUFPS Until-Forward Pixel Seek

Format

- LIB Shell: 0x002C sy xstart xend
- C Binding: void krhwufps(sy, xstart, xend);
- TMLS Interpreter: krhwufps sy xstart xend

Parameters

- short sy: the vertical coordinate of the line to be searched.
- short xstart: the horizontal start coordinate for the search.
- short xend: the horizontal limit coordinate for the search.

Description

This function is used to implement a pixel-seek operation. It seeks a preset seek color along a horizontal screen line, seeking to the right (forwards). The LIB Shell ANDs each pixel with the current seek mask before comparing it to the seek color. The seek continues until either the seek color is found or *zend* is reached without finding the seek color.

Until-Forward Pixel Seek KRHWUFPS

Return Value

The horizontal coordinate of the first pixel satisfying the search condition along the forward search direction. If the search limit was reached before the search terminated, the value returned is one pixel past the limit. The host reads the data as follows:

- LIB Shell: Read data from the Read FIFO entrance.
- C Binding: Use the C binding's iogetw function to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also krhwwfps, krhwwbps, krhwubps, krhwssmsk, krhwsscol

KRHWWBPS While-Backwards Pixel Seek

Format

- LIB Shell: 0x002B sy xstart xend
- C Binding: void krhwwbps(sy, xstart, xend);
- TMLS Interpreter: krhwwbps sy xstart xend

Parameters

- short sy: the vertical coordinate of the line to be searched.
- short xstart: the horizontal start coordinate for the search.
- short xend: the horizontal limit coordinate for the search.

Description

This function is used to implement a pixel-seek operation. It seeks to the left (backwards) along a horizontal screen line while it finds the preset seek color. The LIB Shell ANDs each pixel with the current seek mask before comparing it to the seek color. The seek continues until either the seek color is not found or it reaches xend.

While-Backwards Pixel Seek KRHWWBPS

Return Value

The horizontal coordinate of the first pixel not satisfying the search condition along the backwards search direction. If the search limit was reached before the search terminated, the value returned is one pixel past the limit. The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the iogetw function of the C Binding to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also krhwufps, krhwwfps, krhwubps, krhwssmsk, krhwsscol

KRHWWFPS While-Forward Pixel Seek

Format

- LIB Shell: 0x002A sy xstart xend
- C Binding: void krhwwfps(sy, xstart, xend);
- TMLS Interpreter: krhwwfps sy xstart xend

Parameters

- short sy: the vertical line coordinate to be searched.
- short xstart: the horizontal start coordinate to be searched.
- short xend: the horizontal limit coordinate of the search.

Description

This function is used to implement a pixel-seek operation. It seeks to the right (forwards) along a horizontal screen line while it finds the preset seek color. The LIB Shell ANDs each pixel with the current seek mask before comparing it to the seek color. The seek continues until either the seek color is not found or it reaches xend.

Return Value

The horizontal coordinate of the first pixel not satisfying the search condition along the forward search direction. If the search limit was reached before the search terminated, the value returned is one pixel past the limit. The host reads the data as follows:

While-Forward Pixel Seek KRHWWFPS

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the iogetw function of the C Binding to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also krhwufps, krhwwbps, krhwubps, krhwssmsk, krhwsscol

KRIOSSMODE — Set Swap Mode

Format

- LIB Shell: 0x0012 swapmode
- C Binding: void kriossmode(swapmode);
- TMLS Interpreter: kriossmode swapmode

Parameters

- short swapmode: (Default = 0) the raster data swap mode in the low byte and the general data swap mode (ie: non-raster data) in the high byte. There are modes to swap bytes within a word and/or swap bits within a byte. The following table illustrates how swapping is performed:

INPUT:

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

OUTPUT:

(Mode 0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

(Mode 1)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(Mode 2)

7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

(Mode 3)

8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7

Set Swap Mode — KRIOSSMODE

Description

This function selects the current swap mode for the Write FIFO. The raster data swap modes affect how the shell fetches bit map data defined by the following commands: lsdfont, lsdgc, lsdapat, and lsfbm. The LIB Shell uses the general swap mode when reading other types of data from the Write FIFO.

Return Value None.

Error None.

See Also

LSBMARMSK Select Area Fill Pattern Mask

Format

- LIB Shell: 0x011F spx spy dpx dpy sizex sizey
- C Binding: void lsbmarmsk(spx, spy, dpx, dpy, sizex, sizey);
- TMLS Interpreter: lsbmarmsk spx spy dpx dpy sizex sizey

Parameters

- short spx: the x-coordinate of the top left corner of the desired bit map with respect to the top left corner of the current bit map storage area.
- short spy: the y-coordinate of the top left corner of the desired bit map with respect to the top left corner of the current bit map storage area.
- short dpx: the x-coordinate of the top left corner of the destination area on the screen.
- short dpy: the y-coordinate of the top left corner of the destination area on the screen.
- short sizex: the width of the area to be copied. †
- short sizey: the height of the area to be copied. †

† Setting sizex or sizey to -1 will make the LIB Shell use the full size of the bit map, as defined in its data structures. Other negative values will be ignored.

Select Area Fill Pattern Mask LSBMARMSK

Description

This function uses the selected bit map as a fill mask against the currently selected area fill pattern. The dimensions of the bit map must be greater than the size of the area to be filled. The fill mask is applied as follows: a one bit means draw the pixel (follow the fill pattern) and a zero bit means do not draw the pixel (do not overwrite the original pixel value). Contrast this function with the krhwstdef function.

Note

The lsbmarmsk command cannot be used for general fill operations such as filled circles, ellipses, etc. This function simply copies the current bitmap to the screen using the zeroes and ones in the bitmap as a mask against the currently selected area fill pattern. Thus, ones in the bitmap will be copied to the screen in the color of the currently selected area fill pattern at that point. Zeroes in the bitmap simply cause nothing to be drawn at that point. Though this function is intended for monochrome bitmaps, both monochrome and polychrome area fill patterns can be used.

Return Value None.

Error None.

See Also lssbm, lsafapat, krhwstdef

LSCBM Create Bit Map Storage Area

Format

- LIB Shell: 0x0128 bm_id bm_spec
- C Binding: void lscbm (bm_id, bm_specptr);
- TMLS Interpreter: lscbm bm_id bm_spec

Parameters

- short bm_id: (Default = -1) the label to assign to the bit map storage area. If you attempt to create an area with a label which is currently defined then the previous definition is deleted and redefined. Negative labels are reserved.
- BITMAP bm_spec: the bit map storage area specification. The value of *BITMAP.tl* is ignored. The pitch and format must be set as follows:

	Monochrome	Polychrome
pitch (words)	non-zero	zero or non-zero
format	0x0101	0x0108

The width of polychrome bit maps must be set such that (width × #bits/pixel) is a multiple of 16. A screen depth of 8 bits/pixel (4 bits/pixels) means that widths must be a multiple of 2 (4). The pitch of a polychrome bit map tells the shell how and where bit map data is stored:

- zero: read polychrome bit map data from the screen or from the Write FIFO. If the host sends bit map data via the Write FIFO [1] it must be compressed and [2] each line of the bit map must not be padded.

Create Bit Map Storage Area LSCBM

- non-zero: read bit map data from the Write FIFO. Each line of bit map data is compressed, but may or may not be padded. For example, a 32-bit CPU might send a 2 pixel × 2 pixel bit map as two 32-bit words:

First Line of Bit Map:

Padding	$Pixel_{i+1}$	$Pixel_i$
31		0

Second Line of Bit Map:

Padding	$Pixel_{i+3}$	$Pixel_{i+2}$
31		0

The command to create this bit map storage area is: lscbm <id> 2 2 2 0x0108.

- BITMAP *bm_specptr: a pointer to the bit map specification.

Description

This function allocates an area of Local RAM for storing bit maps. The dimensions of the bit map specification determine the amount of memory the memory manager allocates. Note that you can only store monochrome bit maps in storage areas defined as monochrome, and similarly for polychrome bit maps.

You can copy a bit map from a buffer or from the screen to the storage area. A reverse operation, copying a bit map from the storage area to the screen is also supported.

LSCBM Create Bit Map Storage Area

Return Value None.

Error MEMORY_ERROR, UNSUPPORTED_ERROR (□width specified is not a multiple of two or □ format specified is not 0x0101 or 0x0108)

See Also lssbm, lsrbm, lsfbm, lscpbmms, lscpbmsm

Append to Command List LSCLAPPEND

Format

- LIB Shell: 0x0007 command_list_num execute_flag
- C Binding: void lsclappend(command_list_num, execute_fla
- TMLS Interpreter: lsclappend command_list_num execute_flag

Parameters

- unsigned short command_list_num: The label of the command list to which the data is appended. If the command list specified is not defined then the command is ignored.
- BOOL execute_flag: A flag which indicates whether the command list should be executed during construction:
  - null: Do not execute the command list while it is being constructed.
  - non-null: Execute the command list while it is being constructed.

Description

This function appends data to an existing command list until the next lsclclose command is encountered. The execute_flag parameter allows you to select whether or not the command list is executed as you append data to it.

When the shell appends data to a command list it first tries to append data to the last block of the command list. When no more room remains in this block, it allocates another block of size = current block size set by the lsclsblksz

LSCLAPPEND Append to Command List

command. It continues to allocate memory blocks until either the command list is built or no memory remains.

Use the lsclrealloc command to indicate whether the memory manager reallocates unused memory remaining in the block when the command list is closed.

Note If there is insufficient memory to store the command list being defined, the shell generates a MEMORY_ERROR error. It generates the error as soon as it runs out of memory and deletes the command list. It then halts command list execution (if selected) and starts regular execution of commands from the Write FIFO.

Return Value None.

Error MEMORY_ERROR

See Also lsclclose, lsclsblksz, lsclrealloc, lsclopen

Close Command List — LSCLCLOSE

Format

- LIB Shell: 0x0002
- C Binding: void lsclclose();
- TMLS Interpreter: lsclclose

Parameters None.

Description

This function will close the command list currently being defined. Depending on the option selected with the lsclrealloc command, the memory manager may or may not reallocate any memory not used by the command list definition. If no command list is currently open this command will be ignored.

Return Value None.

Error None.

See Also lsclopen, lsclsblksz, lsclrealloc

LSCLCOMPR Command List Compress

Format

- LIB Shell: 0x148 cl_id
- C Binding: void lsclcompr(cl_id);
- TMLS Interpreter: lsclcompr cl_id

Parameters

- unsigned short cl_id: The 16-bit command list descriptor.

Description

This command compresses a command list by removing any "holes" (wasted memory blocks) left by the lstgrobj and lstgrlay commands. This reclaimed memory is returned to the common memory pool.

Return Value None

Error None

See Also lstgrobj, lstgrlay

Delete Command List — LSCLDEL

Format

- LIB Shell: 0x0005 command_list_num
- C Binding: void lscldel(command_list_num);
- TMLS Interpreter: lsclde1 command_list_num

Parameters

- short command_list_num: the label of the command list to be deleted. If the command list specified is not defined then the command is ignored. A value of -1 tells the LIB Shell to delete all command lists currently defined.

Description

This function deletes the specified command list. The memory manager reclaims the memory occupied by the command list definition.

Return Value None.

Error None.

See Also lsclopen, lsclclose

LSCLEXEC Command List Execute

Format

- LIB Shell: 0x0003 command_list_num
- C Binding: void lsclexec(command_list_num);
- TMLS Interpreter: lsclexec command_list_num

Parameters

- unsigned short command_list_num: the label of the command list to be executed. If the command list specified is not defined then the command is ignored.

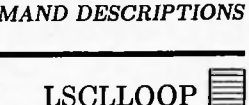
Description

This function executes the specified command list.

Return Value None.

Error CLNEST_ERROR

See Also lsclopen, lsclclose, lsclloop

Command List Loop — LSCLLOOP 

Format

- LIB Shell: 0x0004 command_list_num number_times
- C Binding: void lsclloop(command_list_num, number_times);
- TMLS Interpreter: lsclloop command_list_num, number_times

Parameters

- unsigned short command_list_num: the label of the command list to be executed. If the command list specified is not defined then the command is ignored.
- short number_times: the number of times the command list is to be executed.

Description

This function executes the specified command list number_times times.

Return Value None.

Error CLNEST_ERROR

See Also lsclopen, lsclclose, lsclexec

LSCLMODIFY Modify Command List

Format

- LIB Shell: 0x0008 command_list_num word_offset num_words words
- C Binding: void lsclmodify(command_list_num, word_offset, num_words, wordptr);
- TMLS Interpreter: lsclmodify command_list_num word_offset num_words words

Parameters

- unsigned short command_list_num: the label of the command list to be modified. If the command list specified is not defined then the command is ignored.
- long word_offset: the offset in words from the start of the command list to the first word to be modified.
- long num_words: the number of words of data to be replaced.
- short words[]: the array containing the replacement data. The number of words in words must equal the number of words being replaced. If the replacement data contains less than num_words words then insert lsnoop (no operation) commands to make up the difference or use an lsskip command.
- short *wordptr: a pointer to the buffer containg the replacement data.

Description

This function allows you to replace a section of an existing command list with a new block of data of equal size. You cannot append words to a command list with this command (see lsclappend).

Modify Command List LSCLMODIFY

Note This function works on command lists defined with blk_size = 0, or command lists defined using a block length large enough to house the entire command list.

The lsclmodify command does not work on command lists which are composed of more than one block.

Return Value None.

See Also lsclappend,lsclsblksz

LSCLOPEN Open Command List

Format

- LIB Shell: 0x0001 command_list_num execute_flag
- C Binding: void lsclopen(command_list_num, execute_flag);
- TMLS Interpreter: lsclopen command_list_num execute_flag

Parameters

- unsigned short command_list_num: the label of the command list to be opened. If the command list label specified is already defined then the old version is deleted before the new one is defined.
- BOOL execute_flag: a flag which indicates whether the command list should be executed during construction:
  - null: Do not execute the command list while it is being constructed.
  - non-null: Execute the command list while it is being constructed.

Description

This function starts a command list definition. Each definition must be closed with a lsclclose command. The execute_flag parameter determines whether commands are executed as they are stored in command list memory. Note the description of the command list icons at the beginning of this chapter.

The memory manager allocates one block of Local RAM to store the command list definition. If this memory is not sufficient then it allocates another memory block. It continues to allocate memory blocks until either the command list is built or no memory remains.

Open Command List — LSCLOPEN

The size of the memory blocks is set by the lsclsblksz command. Use the lsclrealloc command to select whether or not the memory manager reallocates unused memory remaining in the block when the command list is closed.

Note If there is insufficient memory to store the command list being defined, the shell will close the command list at the time of the error, and return a CLMEMORY_ERROR.

Return Value None.

Error MEMORY_ERROR

See Also lsclappend, lsclclose, lsclsblksz

LSCLPOS Command List Position

Format

- LIB Shell: 0x000C
- C Binding: void lsclpos();
- TMLS Interpreter: lsclpos

Parameters None.

Description

Use this function during command list building to return the current offset. It helps you keep track of the locations of commands you want to modify later.

Note The offset returned can only be at a command boundary. That is, the lsclpos command cannot be sent in the middle of another command's parameters.

Return Value The offset value is sent to the Read FIFO as a long word. The host reads the data as follows:

- LIB Shell: Read data from the Read FIFO entrance.
- C Binding: Use the lsgetbuf function of the C Binding to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

The lsclpos command returns -1 if no command list is open or if used in a command list which occupies more than one block of memory.

Error None.

See Also lsclmodify, lsclsblksz

Read Command List — LSCLREAD

Format

- LIB Shell: 0x0006 command_list_num
- C Binding: void lsclread(command_list_num);
- TMLS Interpreter: lsclread command_list_num

Parameters

- unsigned short command_list_num: the label of the command list to be read.

Description

This function reads the specified command list and writes the data to the Read FIFO.

Return Value

The first long word is a count of the number of words in the command list. The rest of the buffer contains the contents of the specified command list. Values are hexadecimal. The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the iogetbuf function of the C Binding to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also lsclopen, lsclclose

LSCLREALLOC Cmd. List Reallocate Flag

Format

- LIB Shell: 0x000A realloc_flag
- C Binding: void lsclrealloc (realloc_flag);
- TMLS Interpreter: lsclrealloc realloc_flag

Parameters

- BOOL realloc_flag: (Default = non-null) the memory reallocate flag:
  - null: Do not reallocate memory when the user closes a command list definition.
  - non-null: Reallocate memory when the user closes a command list definition.

Description

This function tells the memory manger whether unused memory is to be reallocated when you close command list definitions.

When a command list is opened, the memory manager allocates blocks of memory whose size is set by the lsclsblksz command. When the command list is closed, any memory left over in the current block is either reallocated to the memory manager (realloc_flag = non-null) or remains allocated to the command list (realloc_flag = null).

The primary use of this command is in conjunction with the lsclappend command. If you know that you will be appending data to a command list, you should use the second option to speed up future additions to the command list.

Cmd. List Reallocate Flag LSCLREALLOC

Return Value None.

Error None.

See Also lsclappend

LSCLSBLKSZ Set Command List Block Size

Format

- LIB Shell: 0x0009 Blk_size
- C Binding: void lsclsblksz(blk_size);
- TMLS Interpreter: lsclsblksz blk_size

Parameters

- unsigned long blk_size: (Default = 20 Kwords) the command list block size in words.

Description

The lsclsblksz command sets the command list block size to blk_size Kwords. Setting blk_size to zero means that the memory manager allocates all available memory for command list building. In this case, set the memory reallocation flag so the memory manager reallocates any unused memory after closing the command list.

Selecting both the allocate-all-memory and the execute-while-build option means the shell will not correctly execute commands which require memory.

Selecting both the allocate-all-memory and the do-not-reallocate-memory option means the memory manager allocates all available memory and does not return unused memory at command list close time.

Note The user can not change the block size during command list building or appending. **Remember that the lsclclose command itself is counted as a word.**

Return Value None.

Error None.

See Also lsclopen, lsclrealloc

Memory Bit Map Copy LSCPBMMS

Format

- LIB Shell: 0x012C spx spy dpx dpy sizex sizey
- C Binding: void lscpbmms(spx, spy, dpx, dpy, sizex, sizey);
- TMLS Interpreter: lscpbmms spx spy dpx dpy sizex sizey

Parameters

- short spx: the x-coordinate of the top left corner of the desired bit map with respect to the top left corner of the current bit map storage area.
- short spy: the y-coordinate of the top left corner of the desired bit map with respect to the top left corner of the current bit map storage area.
- short dpx: the x-coordinate of the top left corner of the destination area with respect to the top left corner of the screen.
- short dpy: the y-coordinate of the top left corner of the destination area with respect to the top left corner of the screen.
- short sizex: the width of the area to be copied. †
- short sizey: the height of the area to be copied. †

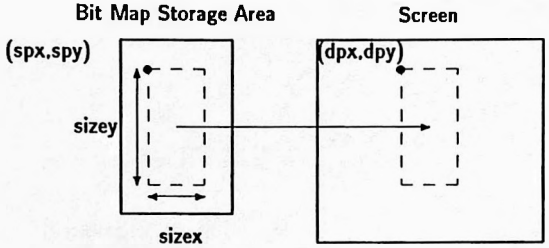
† Setting sizex or sizey to -1 will make the LIB Shell use the full size of the bit map, as defined in its data structures. Other negative values will be ignored.

Description

This function copies a bit map from the current bit map storage area to the screen. The source bit map's top left

LSCPBMMS Memory Bit Map Copy

corner is at (spx, spy) with respect to the top left corner of the bit map storage area; its dimension are sizex × sizey. The bit map is copied onto the screen starting at the point (dpx, dpy).



Return Value None.

Error None.

See Also lscpbmsm, lscbm, lssbm

Screen Bit Map Copy LSCPBMSM

Format

- LIB Shell: 0x012B spx spy dpx dpy sizex sizey
- C Binding: void lscpbmsm(spx, sdy, dpx, dpy, sizex, sizey);
- TMLS Interpreter: lscpbmsm spx spy dpx dpy sizex sizey

Parameters

- short spx: the x-coordinate with respect to the top left corner of the screen area to copy into memory.
- short spy: the y-coordinate with respect to the top left corner of the screen area to copy into memory.
- short dpx: the x-coordinate with respect to the top left corner of the current bit map storage area where the specified screen contents will be stored.
- short dpy: the y-coordinate with respect to the top left corner of the current bit map storage area where the specified screen contents will be stored.
- short sizex: the width of the area to be copied. †
- short sizey: the height of the area to be copied. †

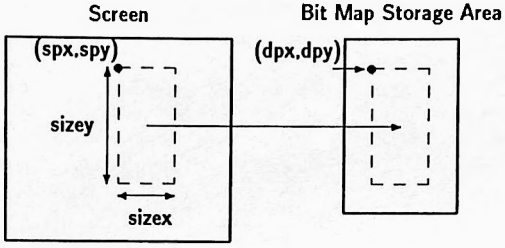
† Setting sizex or sizey to -1 will make the LIB Shell use the full size of the bit map, as defined in its data structures. Other negative values will be ignored.

Description

This function copies a polychrome bit map from the screen to the current bit map storage area. The current storage area must be a polychrome one. The source bit map's top

LSCPBMSM Screen Bit Map Copy

left corner is at (spx, spy); its dimension are sizex × sizey. The bit map is copied into the current bit map storage area starting at the point (dpx, dpy) with respect to the top left corner of the area.



Note Use the lsfbm function to fill a monochrome bit map storage area. Note that you only store monochrome bit maps in monochrome bit map storage areas.

Return Value None.

Error UNSUPPORTED_ERROR (□ can not copy from the screen to a monochrome bit map storage area)

See Also lscbm, lsfbm, lssbm

Copy Bitmap With Zoom LSCPBMZM

Format

- LIB Shell: 0x011E spx spy dpx dpy sizex sizey zoomx zoomy
- C Binding: void lscpbmzm(spx,spy,dpx,dpy,sizex,sizey, zoomx,zoomy);
- TMLS Interpreter: lscpbmzm spx spy dpx dpy sizex sizey zoomx zoomy

Parameters

- short spx: x-coordinate with respect to the top left corner of the current bitmap storage area.
- short spy: y-coordinate with respect to the top left corner of the current bitmap storage area.
- short dpx: x-coordinate of the destination area on the screen, with respect to the top left corner of the screen.
- short dpy: y-coordinate of the destination area on the screen, with respect to the top left corner of the screen.
- short sizex: x-size of the source to copy. †
- short sizey: y-size of the source to copy. †
- FIXPT zoomx: x-axis zoom factor.
- FIXPT zoomy: y-axis zoom factor.

† Setting sizex or sizey to -1 will make the LIB Shell use the full size of the bit map, as defined in its data structures.

Define Fill Pattern LSDAPAT

Format

- LIB Shell: 0x0058 fillpat_id fillpat_spec
- C Binding: void lsdapat(fillpat_id, fillpat_ptr);
- TMLS Interpreter: lsdapat fillpat_id fillpat_spec

Parameters

- short fillpat_id: the label assigned to the fill pattern to define. The LIB Shell first attempts to delete the fill pattern (in case it already exists), and then proceeds with the definition. Attempts to use a negative fillpat_id will result in all user-defined bit maps being deleted, and the current definition being ignored.
- BITMAP fillpat_spec: the fill pattern specification. You can define both polychrome and monochrome fill patterns with this command. The LIB Shell distinguishes the type of fill pattern by the value of the pitch and format elements of BITMAP. Parameters are set as follows:

Parameter	Monochrome	Polychrome
width (pixels)	1-65535	1-65535
heigth (pixels)	1-65535	1-65535
pitch (words)	non-zero	0
format	0x0101	0x0108

A monochrome bit map is sent to the LIB Shell in a buffer. A polychrome bit map is copied from the screen. For this reason the *tl* element is interpreted in one of several ways, depending on the type of bit map and how you are executing the command:

LSCPBMZM Copy Bitmap With Zoom

Description

This command is used to zoom a rectangular area of the current monochrome or polychrome bit map into the frame buffer. The source bitmap is the one currently selected by the lssbm command. Screen-to-screen zooming can be achieved by using the frame buffer as the currently selected bitmap (ID = -1). In such a case, the user should be careful to specify non-overlapping source and destination areas.

Return Value None.

Error None.

See Also lscbm, lssbm, lsfbm, lscpbmsm, lscpbmms.

LSDAPAT — Define Fill Pattern

Software	Monochrome	Polychrome
LIB Shell & TMLS	the bit map data buffer	(x,y) screen coordinate
C Binding	a pointer to the bit map data buffer	a pointer to (x,y) screen coordinate

The (x,y) screen coordinate is the top left corner of the width × height area you are selecting. **Note that the x-coordinate must be a multiple of 8.**

- BITMAP *fillpat_ptr: a pointer to the fill pattern specification.

Description

This function allows the user to define a fill pattern. Fill patterns are used in area fills and wide line fills. The lssafapat command selects the current area fill pattern, while lsswlapat command selects the current wide line fill pattern. A fill pattern definition must be selected in order to affect subsequent drawing operations.

Note We recommend that you define polychrome bit maps in hidden screen memory. This should reduce the chance of accidently overwriting part of your pattern and corrupting it. Unlike monochrome fill patterns, polychrome fill patterns are **not** stored in Local RAM.

Return Value None.

Error MEMORY_ERROR, UNSUPPORTED_ERROR
(□ x-coordinate specified is not a multiple of 8 or □ illegal format specified)

See Also lssafapat, lsrapat, lsswlapat, scarspin

Define Clipping List — LSDCLPL

Format

- LIB Shell: 0x0136 id num_clipr $clipr_1$ . . . $clipr_{num\_clipr}$
- C Binding: void lsdclpl (id num_clipr, clipptr);
- TMLS Interpreter: lsdclpl id num_clipr $clipr_1$ . . . $clipr_{num\_clipr}$

Parameters

- short id: the label of the clipping list being defined. If you attempt to define a clipping list with a label that has already been defined, the shell will delete the previous definition and redefine it. A negative label indicates that all user-defined clipping lists are to be removed, and the current definition ignored.
- short num_clipr: the number of clipping rectangles in the clipping list.
- short clipr[]: the array containing the coordinates of *num_clipr* clipping rectangles. Coordinates for each clipping rectangle must be specified in the following order: left, top, right, and bottom.
- short *clipptr: a pointer to the array containing the clipping rectangle coordinates.

Description

This function is used to define a multi-clipping rectangle list. The LIB Shell stores the list in Local RAM.

Return Value None.

Error None.

See Also lssclpl, lsrclpl

LSDFONT — Define Text Font

Format

- LIB Shell: 0x0065 font_id font_spec
- C Binding: void lsdfont(font_id, font_ptr);
- TMLS Interpreter: lsdfont font_id font_spec

Parameters

- short font_id: the label assigned to the font. If you attempt to define an font with a font_id that has already been defined then the previous font definition is deleted and redefined. A negative label indicates that all user-defined fonts are to be removed, and the current definition ignored.
- FONT font_spec: the font specification. Refer to Chapter 2 for a description of the FONT structure.
- FONT *font_ptr: a pointer to the font specification.

Description

This function is used to define fonts. The font may be mono-spaced or proportionally spaced, raster or stroke, and may contain kerning pairs. Once a font has been defined it must be selected in order to affect subsequent text drawing.

Return Value None.

Error MEMORY_ERROR

See Also lssfont, lsrfont

Define Cursor Shape — LSDGC

Format

- LIB Shell: 0x007F cursor_id xhot yhot gc_bmap
- C Binding: void lsdgc(cursor_id, xhot, yhot, gc_bmapptr);
- TMLS Interpreter: lsdgc cursor_id xhot yhot gc_bmap

Parameters

- short cursor_id: the label assigned to the graphic cursor definition. If you attempt to define a graphics cursor with a label which is already used then the previous definition will be deleted and redefined. A negative label indicates that all user-defined graphics cursors are to be removed, and the current definitions ignored.
- short xhot: the x-coordinate of the cursor hot-spot relative to the top left corner of the cursor bit map.
- short yhot: the y-coordinate of the cursor hot-spot relative to the top left corner of the cursor bit map.
- BITMAP gc_bmap: the graphics cursor bit map specification. The cursor bit map contains two bit maps of equal size. The top bit map is used in the first drawing pass while the bottom bit map is used for the second drawing pass. Thus, the cursor bit map consists of 2 × (height × pitch) words in total. The format must be 0x0101.

  For speed considerations **we restrict the pitch of cursor bit maps as follows:** the pitch must be a power of 2. For example, a pitch of 3 words can not be used.

LSDGC Define Cursor Shape

For boards that have hardware cursor capability, use the krgcsshp command to define the user-defined cursor shape.

(Boards using a hardware cursor only) The cursor controller chip can only handle user-defined graphics cursor shapes which are 64 pixels wide by 64 pixels high. See the description of the krgcsshp command in your supplemental manual for more information.

- BITMAP *gc_bmapptr: a pointer to the graphics bit map specification.

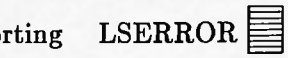
Description

This function allows the user to define a graphics cursor. Once defined, a cursor must be enabled and selected in order to affect subsequent cursor drawing operations.

Return Value None.

Error MEMORY_ERROR, UNSUPPORTED_ERROR (□pitch is not a power of 2 or □illegal format specified)

See Also lssgc, lsrgc, krgcen, krgcdi, krgcsdm, krgcsda

Enable/Disable Error Reporting LSERROR 

Format

- LIB Shell: 0x0013 flag
- C Binding: void lserror(flag);
- TMLS Interpreter: lserror flag

Parameters

- BOOL flag: the flag to enable/disable error reporting:
  - null: Disable error reporting.
  - non-null: Enable error reporting.

Description

This function selects whether error messages are reported in the ERROR register.

Note

Error reporting is only performed when the user enables this feature with the lserror command. Once enabled, the shell returns an 8-bit error message in the ERROR Register each time an error occurs. Note that if the LIB Shell generates an error message, the system will wait for the user to read the current error message before reporting the next error. The system will appear to hang if errors are not removed from the register as they occur.

The LIB Shell currently detects the following error types:

- OPCODE_ERROR: The user sent an undefined opcode to the shell.
- UNSUPPORTED_ERROR: The user sent an unsupported parameter.

LSERROR Enable/Disable Error Reporting

- MEMORY_ERROR: The LIB Shell ran out of memory while defining an area pattern, a graphics cursor, a font, or a bit map.
- OVERFLOW_ERROR: An arithmetic overflow occurred.
- CLNEST_ERROR: The user nested command list executions too deep.
- CLMEMORY_ERROR: The LIB Shell ran out of memory at command list build time.

Error codes are returned to the ERROR register as follows:

Bit	Error
0	OPCODE_ERROR
1	UNSUPPORTED_ERROR
2	MEMORY_ERROR
3	OVERFLOW_ERROR
4	CLNEST_ERROR
5	CLMEMORY_ERROR

Figure 3.1: The ERROR Register

Return Value None.

Error None.

See Also

Fill Bit Map Storage Area — LSFBM

Format

- LIB Shell: 0x012A bm_data
- C Binding: void lsfbm(bm_length, bm_dataptr);
- TMLS Interpreter: lsfbm bm_data

Parameters

- unsigned short bm_data []: a buffer containing the bit map data. Pixel data is compressed as follows:

8 bits/pixel:

$Pixel_{i+1}$	$Pixel_i$

- unsigned short *bm_dataptr: a pointer to the buffer containing the bit map data.
- short bm_length: the length of the data buffer in words.

Description

This function allows you to fill the current bit map storage area with the data contained in buffer bm_data. Note that the amount of data (bm_length) in the buffer must be enough to fill the entire storage area.

Return Value None.

Error None.

See Also lssbm, lscpbmsm, lscpsmms

LSINQ — Inquire Shell Parameters

Format

- LIB Shell: 0x0130 inq_index
- C Binding: void lsinq(inq_index);
- TMLS Interpreter: lsinq inq_index

Parameters

- unsigned short inq_index: the index of the LIB Shell parameter being queried. See the tables below.

inq_index	LIB Shell Parameter	# Words	Format
	Cursor		
31	cursor enable/disable status	1	krgcen/krgcdi
32	cursor position	2	krgcmv
33	cursor drawing attributes	11	krgcsda
82	cursor shape	1	lssgc
87	cursor display mode	1	krgcsdm
	General Drawing Attributes		
34	background color	1	krhwsbcol
35	boolean operator	1	krhwsbop
36	clipping rectangle	4	krhwsclipr
38	foreground color	1	krhwsfcol
40	plane mask	1	krhwsmsk
43	transparency mode	1	krhwstdef
50	theta increment angle	1	scsthincr

Inquire Shell Parameters — LSINQ

inq_index	LIB Shell Parameter	# Words	Format
	Outline Attributes		
37	current line style	2	krhwscls
39	initial line style	2	krhwsils
46	endpoint shape	3	scolseshp
47	outline mode	1	scolsmode
48	outline width	1	scolsw
83	wide line fill pattern	1	lsswlapat
	Pixel Seek		
41	pixel seek color	1	krhwsscol
42	pixel seek mask	1	krhwssmsk
	Current Points		
45	lineto-moveto position	2	scolmvto
67	current 2D point	4	v2ollnto
30	current 3D point	6	v3olmvto
	Text		
51	alignment options	2	sctxsalign
52	boldening factor	1	sctxsbf
53	text baseline rotation angle	1	sctxsblrot
54	interword spacing	2	sctxsbx
55	character cell rotation angle	1	sctxschrot
56	text drawing mode	1	sctxsdmode
57	italicization mask	1	sctxsim
58	intercharacter spacing	1	sctxsincs
59	lightening mask	1	sctxslm
60	text scaling factors	4	sctxssf
61	strike-out thickness	1	sctxssow
62	stroke width	1	sctxssw
63	underline width	1	sctxsuw
64	word placement width	1	sctxswpw
81	current font label	1	lssfont

LSINQ — Inquire Shell Parameters

inq_index	LIB Shell Parameter	# Words	Format
	Graphics Transformations		
49	screen origin	2	scsorg
65	viewport	4	v2gpsvwp
66	window	8	v2gpswdw
68	hither plane distance	2	v3gpsdisth
69	yon plane distance	2	v3gpsdisty
70	hither clipping flag on/off	1	v3gpshclip
71	yon clipping flag on/off	1	v3gpsyclip
72	viewing angle	1	v3gpsproj
27	modelling matrix (FIXPT)	4×4×2	v3mdset
28	viewing matrix (FIXPT)	4×4×2	v3vwset
73	viewing reference point	6	v3vwrpt
74	modelling origin	6	v3mdorg
	Others		
44	Write FIFO data swap mode	1	kriossmode
75	command list block size	2	lsclsblksz
76†	EGA enabled/disabled	1	
77	error reporting status on/off	1	lserror
78	current logical LUT	1	lslutsl
79	current area fill pattern	1	lssafapat
80	current bit map storage area	1	lssbm
84	available # words of Local RAM	2	lsinqmem
85	system parameters	10	lsinqsys
86†	EGA-1281 present/not present	1	$\neq 0$ if present

† — only applies to PG-Series-1 users.

Inquire Shell Parameters — LSINQ

inq_index	LIB Shell Parameter	# Words	Format
	Others		
88	physical LUT 0 plane mask	1	lslutpm
89	physical LUT 1 plane mask	1	lslutpm
90	amount of Local RAM (in words) required for CGA emulation. Invoke lsemulen with save restore flag enabled.	1	lsemulen
91	physical size of Local RAM	2	none
92	spinoff characteristics #1: overlay width #2: overlay height #3: overlay depth #4: presence = 1, absence = 0, of: bit 0: hardware cursor controller bit 1: serial controller bit 2: zoom/pan gate array #5: number of ROGAs	5	none
93	zoom/pan factors	6	krhwszp

Description

This function allows the user to inquire the current status of flags, attributes, matrices, and other LIB Shell parameters.

Return Value

The number of words returned depends on the parameter of interest. The first word returned contains a count of the number of words left to fetch from the Read FIFO for that inquiry. The host reads the data as follows:

- LIB Shell: Read data from the Read FIFO entrance.

LSINQ Inquire Shell Parameters

- C Binding: Use the iogetbuf function of the C Binding to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

If an illegal inq_index is specified, a zero is returned in the report FIFO to indicate that there are no words to fetch.

Error None.

See Also lsinqgcp, lsinqmem, lsinqsys

Inquire Graphics Cursor Position LSINQGCP

Format

- LIB Shell: 0x0087
- C Binding: void lsinqgcp();
- TMLS Interpreter: lsinqgcp

Parameters None.

Description

This command returns the graphic cursor current position in screen coordinates. The x-coordinate is returned first, then the y-coordinate.

Return Value

The cursor position is sent to the Read FIFO as two short words. The host reads the data as follows:

- LIB Shell: Read data from the Read FIFO entrance.
- C Binding: Use the iogetbuf function of the C Binding (or iogetw twice) to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also krgcmv, lsinq

LSINQMEM Inquire Available Memory

Format

- LIB Shell: 0x000B
- C Binding: void lsinqmem();
- TMLS Interpreter: lsinqmem

Parameters None.

Description

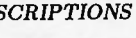
This function returns the amount of Local RAM available for command lists, user defined fill patterns, bit map storage areas, etc.

Return Value The number of words available in Local RAM is returned to the Read FIFO as a long word. The host reads the data as follows:

- LIB SHELL: Read data from the Read FIFO entrance.
- C Binding: Use the iogetbuf function of the C Binding to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also lsinqgcp, lsinqsys, lsinq

Inquire System Parameters LSINQSYS 

Format

- LIB Shell: 0x012F
- C Binding: void lsinqsys();
- TMLS Interpreter: lsinqsys

Parameters None.

Description

This function returns the following system level parameters:

Word:	Parameter:	Typical Value:
1	visible screen width in pixels	0x0500
2	visible screen height in pixels	0x0400
3	maximum number of colors that can be simultaneously displayed	0x0100
4	high byte = # of physical LUTs, low byte = DAC size (4 or 8 bit)	0x0108
5	frame buffer width in pixels	0x0800
6	frame buffer height in pixels	0x0400
7	frame buffer pitch in pixels	0x4000
8	frame buffer format: high byte = #planes; low byte = #bits/pixel	0x0108
9	board type	0x0400
10	version stamp	0x0000

LSINQSYS Inquire System Parameters

Return Value

The system parameters listed above. The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the iogetbuf function of the C Binding to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also lsinq, lsinqgcp, lsinqmem

Set LUT Blinking — LSLUTBLINK

Format

- LIB Shell: 0x0126 phy_lut index r g b offtime ontime
- C Binding: void lslutblink(phy_lut, index, r, g, b, offtime, ontime);
- TMLS Interpreter: lslutblink phy_lut index r g b offtime ontime

Parameters

- unsigned short phy_lut: a physical lookup table label. A value of 0 is expected unless there are two physical LUTs (0 and 1).
- unsigned short index: the index of the color being blinked.
- unsigned short r: the red intensity of the blink color.
- unsigned short g: the green intensity of the blink color.
- unsigned short b: the blue intensity of the blink color.
- unsigned short ontime: the amount of time the original color is displayed, expressed in frames. A frame time is $\frac{1}{60}$ of a second.
- unsigned short offtime: the amount of time the blink color is displayed, expressed in frames. A frame time is $\frac{1}{60}$ of a second.

Description

This function causes all pixels drawn in the index index with respect to the current physical LUT to blink to the specified blink color. The ontime is the time the original color is displayed. The offtime is the amount of time the

LSLUTBLINK — Set LUT Blinking

blink color is displayed. You can select up to sixteen blink colors at any given time.

To disable blinking for a particular index simply issue a second lslutblink command, specifying the same physical LUT and index but passing zeroes for the offtime and ontime parameters and dummy RGB values.

Use lslutsblink to disable all blinking.

Return Value None.

Error None.

See Also lslutsblink

Modify Logical LUT LSLUTMD

Format

- LIB Shell: 0x0033 log_lut first_entry nentries $color_1$ . . . $color_{nentries}$
- C Binding: void lslutmd(log_lut, first_entry, nentries, colors);
- TMLS Interpreter: lslutmd log_lut first_entry nentries $color_1$ . . . $color_{nentries}$

Parameters

- unsigned short log_lut: a logical lookup table label; values in the range of 0 to 3 are expected. If this table is currently loaded in a physical LUT then that physical LUT is also modified.
- unsigned short first_entry: the index of the first entry to be modified.
- unsigned short nentries: the number of entries to be modified.
- LUTVAL $color_i$: the new color values $color_i$.
- LUTVAL *colors: a pointer to a buffer containing the new color values.

Description

This function replaces nentries entries in the logical lookup table log_lut with the specified values, starting at entry first_entry.

Return Value None.

Error None.

See Also lslutsl, lslutrd, lslutrs

LSLUTPM Set LUT Plane Mask

Format

- LIB Shell: 0x0125 phy_lut bitmask
- C Binding: void lslutpm(phy_lut, bitmask);
- TMLS Interpreter: lslutpm phy_lut bitmask

Parameters

- unsigned short phy_lut: a physical lookup table label. A value of 0 is expected unless there are two physical LUTs (0 and 1).
- unsigned short bitmask: a value defining the 8-bit output mask. A value in the range of 0 to 255 is expected. Pixel values in the frame buffer are ANDed with the bit mask before they are sent to the current physical Lookup Table. A one bit selects a bit plane, while a zero bit masks it. A bit mask containing all zeroes means data is sent to the output LUT as zeroes.

Description

This function sets the 8-bit output mask to the value contained in bitmask parameter.

Return Value None.

Error None.

See Also lslutsl

Read LUT — LSLUTRD

Format

- LIB Shell: 0x0034 log_lut first_entry nentries
- C Binding: void lslutrd(log_lut, first_entry, nentries);
- TMLS Interpreter: lslutrd log_lut first_entry nentries

Parameters

- unsigned short log_lut: a logical lookup table label; values in the range 0 to 3 are expected.
- unsigned short first_entry: the index of the first entry to be read.
- unsigned short nentries: the number of entries to be modified.
- LUTVAL *data: a pointer to the buffer used to return the lookup table data.

Description

This function reads nentries entries from the logical LUT log_lut, starting at index first_entry, and sends the data to the host.

Return Value

The LIB Shell writes the entries from the specified logical LUT to the Read FIFO. The host retrieves the data as follows:

- LIB Shell: Read data from the Read FIFO entrance.
- C Binding: Use the iogetbuf function of the C Binding to retrieve the data.
- TMLS Interpreter: Use the lsrdfifo utility.

LSLUTRD — Read LUT

Error None.

See Also lslutsl, lslutmd, lslutrs

Reset LUT — LSLUTRS

Format

- LIB Shell: 0x0035 def_lut log_lut
- C Binding: void lslutrs(def_lut, log_lut);
- TMLS Interpreter: lslutrs def_lut log_lut

Parameters

- short def_lut: a default lookup table label. A value of -1 is expected.
- unsigned short log_lut: a logical lookup table label, values in the range of 0 to 3 are expected. If this table is the current LUT then the physical LUT is also modified.

Description

This function resets the logical lookup table log_lut to the values stored in the default lookup table def_lut.

Return Value None.

Error None.

See Also lslutsl, lslutmd, lslutrd

LSLUTSBLINK Stop All Blinking

Format

- LIB Shell: 0x0127
- C Binding: void lslutsblink();
- TMLS Interpreter: lslutsblink

Parameters None.

Description

This function disables all blinking (if enabled). If you only want to stop one color index from blinking refer to the lslutblink command.

Return Value None.

Error None.

See Also lslutblink

Select Logical LUT — LSLUTSL

Format

- LIB Shell: 0x0032 log_lut phy_lut
- C Binding: void lslutsl(log_lut, phy_lut);
- TMLS Interpreter: lslutsl log_lut phy_lut

Parameters

- unsigned short log_lut: (Default = 0) a logical lookup table label; values in the range of 0 to 3 are expected.
- unsigned short phy_lut: (Default = 0) a physical lookup table label. A value of 0 is expected unless there are two physical LUTs (0 and 1).

Description

This function loads a logical lookup table into the selected physical LUT.

Return Value None.

Error None.

See Also lslutmd, lslutrd, lslutrs, lslutpm

LSNOOP — No Operation

Format

- LIB Shell: 0x0122
- C Binding: void lsnoop();
- TMLS Interpreter: lsnoop

Parameters None.

Description

This function does nothing. Its main uses are as a replacement word when editing a command list and FIFO synchronization when using krhwsdirect.

Return Value None.

Error None.

See Also lsclmodify, krhwsdirect, lsclread, lsskip, lsvsint

Disable Picking LSPKDI

Format

- LIB Shell: 0x0132
- C Binding: void lspkdi ():
- TMLS Interpreter: lspkdi

Parameters None.

Description

This function disables picking.

Return Value None.

Error None.

See Also lspken, lspksidl, lspksidh

LSPKEN — Enable Picking

Format

- LIB Shell: 0x0131 x y half_xsize half_ysize
- C Binding: void lspken (x, y, half_xsize, half_ysize);
- TMLS Interpreter: lspken x y half_xsize half_ysize

Parameters

- short x: the x-coordinate of the center of the pick aperture.
- short y: the y-coordinate of the center of the pick aperture.
- unsigned short half_xsize: half the width of the pick aperture.
- unsigned short half_ysize: half the height of the pick aperture.

Description

This function enables pick mode. In this mode you can not draw on the screen. If the LIB Shell detects an attempt to write pixel values within the pick aperture it transmits a pick report to the host. A pick report consists of the current 32 bit pick ID value. The LIB Shell remains in pick mode until the user disables it with the lspkdi command.

The shell will only generate multiple pick reports within the same pick "session" if the current pick ID value to be reported is different from the last value reported. On power up and when picking is enabled, the current pick ID is set to zero and the last reported pick ID is initialized to 0xFFFF 0xFFFF.

Enable Picking — LSPKEN

Notes

- The shell will ignore attempts to redefine the the current clipping rectangle when in pick mode. When you disable picking the shell restores the clipping rectangle active prior to the pick session.
- If you execute the v2gpsvwp command when in pick mode the LIB Shell will recalculate the window to viewport mapping ratios. It will not reset the clipping rectangle to the viewport.
- Picking does not apply to accesses through the 4 Kbyte window or the 5 Mbyte window, only to incoming commands.

Return Value None.

Error None.

See Also lspkdi, lspksidl, lspksidh

LSPKSIDH — Set Pick ID High Word

Format

- LIB Shell: 0x0133 high_id
- C Binding: void lspksidl (high_id);
- TMLS Interpreter: lspksidl high_id

Parameters

- unsigned short high_id: (Default = 0) the high word of the 32 bit pick ID.

Description

This function sets the high word of the current pick ID.

Return Value None.

Error None.

See Also lspken, lspkdi, lspksidl

Set Pick ID Low Word — LSPKSIDL

Format

- LIB Shell: 0x0133 low_id
- C Binding: void lspksidl (low_id);
- TMLS Interpreter: lspksidl low_id

Parameters

- unsigned short low_id: (Default = 0) the low word of the 32-bit pick ID.

Description

This function sets the low word of the current pick ID.

Return Value None.

Error None.

See Also lspken, lspkdi, lspksidh

LSRAPAT Remove Fill Pattern Definition

Format

- LIB Shell: 0x005A apat_id
- C Binding: void lsrapat(apat_id);
- TMLS Interpreter: lsrapat apat_id

Parameters

- short apat_id: the label of a fill pattern definition. If you remove the currently selected area fill pattern definition then subsequent area fills will use the default area fill pattern (label = -1, pattern = solid). The same applies to the current wide line fill pattern and wide line fills. A negative label indicates that all user-defined area fill patterns are to be removed.

Description

This function removes a fill pattern from the current list of defined fill patterns. The memory occupied by the definition is reclaimed by the memory manager.

Return Value None.

Error None.

See Also lsdapat, lssafapat, lsswlapat

Remove Bit Map Storage Area LSRBM

Format

- LIB Shell: 0x012D bm_id
- C Binding: void lsrbm(bm_id);
- TMLS Interpreter: lsrbm bm_id

Parameters

- short bm_id: the label of the bit map storage area to be removed. If the current storage area is removed, then the default bit map storage area -1 (the frame buffer) is selected for subsequent bit map operations. A negative label indicates that all user-defined bit map storage areas are to be removed.

Description

This function removes the specified bit map storage area from the list of defined bit map storage areas. The memory manager reclaims the memory allocated when the storage area was created.

Return Value None.

Error None.

See Also lscbm, lssbm

LSRCLPL Remove Clipping List

Format

- LIB Shell: 0x0138 id
- C Binding: void lsrclpl (id);
- TMLS Interpreter: lsrclpl id

Parameters

- unsigned short id: the label of the clipping list to be removed.

  If you remove the currently selected clipping list the shell selects clip list -1, disabling multi-rectangle clipping, and restores the clipping rectangle set by the krhwsclipr command prior to enabling multi-rectangle clipping. A negative label indicates that all user-defined clipping lists are to be removed.

Description

This function removes a clipping list from the current list of defined clip lists. The memory manager reclaims the area of Local RAM containing the clipping list.

Return Value None.

Error None.

See Also lsdclpl, lssclpl

Read Raster Image — LSRDRST

Format

- LIB Shell: 0x0020 left top right bottom
- C Binding: void lsrdrst(left, top, right, bottom, dest_filename);
- TMLS Interpreter: lsrdrst left top right bottom

Parameters

- short left: the x-coordinate of the top left corner of the screen image.
- short top: the y-coordinate of the top left corner of the screen image.
- short right: the x-coordinate of the bottom right corner of the screen image.
- short bottom: the y-coordinate of the bottom right corner of the screen image.
- char dest_filename[]: binary file where the raster image is copied to.

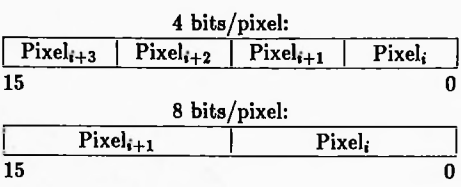
Description

This function reads a screen image from top to bottom, left to right, and transmits the data to the Read FIFO. The extent of the image is determined from the coordinates of the screen rectangle supplied.

Return Value The screen image is returned in the Read FIFO as hexadecimal values.

LSRDRST Read Raster Image

Data is compressed as follows:



The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the iogetbuf function of the C Binding to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also lswrrst, lsrlwrrst, lsrlrdrst

Remove Font Definition — LSRFONT

Format

- LIB Shell: 0x0067 font_id
- C Binding: void sctxrfont(font_id);
- TMLS Interpreter: lsrfont font_id

Parameters

- short font_id: the label of the font definition to be removed. If you remove the currently selected font definition then subsequent text drawing will be performed in the default font (label = -1). A negative label indicates that all user-defined fonts are to be removed.

Description

This function removes a previously-defined font. The function will delete the font having font_id as a label. The memory occupied by the font is reallocated by the memory manager.

Return Value None.

Error None.

See Also lsdfont, lssfont

LSRGC Remove Graphics Cursor Definition

Format

- LIB Shell: 0x0081 cursor_id
- C Binding: void lsrgc(cursor_id);
- TMLS Interpreter: lsrgc cursor_id

Parameters

- short cursor_id: the label of the cursor definition. If you remove the currently selected graphics cursor definition then subsequent cursor operations will be performed with the default graphics cursor (label = -1, shape = arrow cursor). A negative label indicates that all user-defined cursors are to be removed.

Description

This function removes the specified graphic cursor from the current list of defined cursors. The memory occupied by the graphics cursor definition is reallocated by the memory manager.

Return Value None.

Error None.

See Also krgcsdm, lsrgc, lssgc, krgcdi, krgcen, krgcsda

Run Length Raster Read LSRLRDRST

Format

- LIB Shell: 0x0037 left top right bottom
- C Binding: void lsrlrdrst(left, top, right, bottom, dest_filename);
- TMLS Interpreter: lsrlrdrst left top right bottom

Parameters

- short left: the x-coordinate of the top left corner of the screen image.
- short top: the y-coordinate of the top left corner of the screen image.
- short right: the x-coordinate of the bottom right corner of the screen image.
- short bottom: the y-coordinate of the bottom right corner of the screen image.
- char dest_filename[]: binary file where the raster image is copied to.

Description

This function reads a screen image into a run length encoded image. It reads the image going from top to bottom, left to right. The extent of the image is determined from the coordinates of the screen rectangle supplied.

LSRLRDRST Run Length Raster Read

Return Value

A long word is returned containing the number of words in the run length encoded image, following by the run length encoded image. Data is hexadecimal, with the color index in the low byte and the run length in the high byte of each word. The host reads the data as follows:

LIB SHELL: Read data from the Read FIFO entrance.

C Binding: Use the iogetbuf function of the C Binding to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also lsrdrst, lsrlwrrst, lswrrst

Run Length Raster Write LSRLWRRST

Format

- LIB Shell: 0x0036 left top right bottom rl_data
- C Binding: void lsrlwrrst(left, top, right, bottom, src_filename);
- TMLS Interpreter: lsrlwrrst left top right bottom rl_data

Parameters

- short left: the x-coordinate of the top left corner of the screen rectangle.
- short top: the y-coordinate of the top left corner of the screen rectangle.
- short right: the x-coordinate of the bottom right corner of the screen rectangle.
- short bottom: the y-coordinate of the bottom right corner of the screen rectangle.
- short rl_data[]: the buffer containing the run length encoded data. The color index is in the low byte of each word and the run length is in the high byte.
- char src_filename[]: binary file where the raster image is copied from.

Description

This function displays a run length encoded image on the screen, bounded by the specified rectangle. It writes the image from top to bottom, left to right.

Return Value None.

Error None.

See Also lsrdrst, lsrlrdrst, lswrrst

LSSAFAPAT Select Area Fill Pattern

Format

- LIB Shell: 0x0059 apat_id
- C Binding: void lssafapat(apat_id);
- TMLS Interpreter: lssafapat apat_id

Parameters

- short apat_id: (Default = -1) the label of a fill pattern definition. Four pre-defined fill patterns are provided:

apat_id:	fill pattern:
-1	solid
-2	hollow
-3	brick
-4	diagonal lines

If the fill pattern selected is not defined then this command is ignored. On power up fill pattern -1 is selected.

Description

This function selects the fill pattern to be used for subsequent area fills.

Return Value None.

Error None.

See Also lsdapat, lsrapat, scarspin

Select Bit Map Storage Area LSSBM

Format

- LIB Shell: 0x0129 bm_id
- C Binding: void lssbm(bm_id);
- TMLS Interpreter: lssbm bm_id

Parameters

- short bm_id: (Default = -1) the label of a bit map storage area. If the label specified is not defined then the command is ignored.

Description

This function selects the bit map storage area to be used for subsequent bit map operations.

Return Value None.

Error None.

See Also lscbm, lsrbm, lsfbm, lscpbmsm, lscpbmms

LSSCLPL Select Clipping List

Format

- LIB Shell: 0x0137 id
- C Binding: void lssclpl (id);
- TMLS Interpreter: lssclpl id

Parameters

- short id: (Default = -1) the label of a clipping list. Select clipping list -1 to disable multi-rectangle clipping. The shell resets the clipping rectangle to that set by the krhwsclipr command prior to enabling multi-rectangle clipping.

Description

This function selects a clipping list. The lists consists of one or more clipping rectangle coordinates. The LIB Shell will use these clipping rectangles to clip subsequent drawing until the user disables multi-rectangle clipping.

When multi-rectangle clipping is enabled, any attempts to modify the current clipping rectangle using the krhwsclipr command are ignored.

Executing the v2gpsvwp command will result in a recalculation of the window to viewport mapping ratios but will not reset the clipping rectangle.

Multi-rectangle clipping only affects graphics drawn using LIB Shell commands, not direct frame buffer accesses through the 4 Kbyte window or 5 Mbyte windows.

Select Clipping List — LSSCLPL

Return Value None.

Error None.

See Also lsdclpl, lsrclpl

LSSFONT — Select a Font

Format

- LIB Shell: 0x0066 font_id
- C Binding: void lssfont(font_id);
- TMLS Interpreter: lssfont font_id

Parameters

- short font_id: (Default = -1) the label of the selected font. The default font_id selects one of three raster fonts, depending on the resolution of your monitor. For example, if your monitor's resolution is 1280×1024 the shell selects the 16 pixel×32 pixel mono-spaced raster font. The table below shows the labels of the system defined fonts:

font_id	Font
-1	default raster font: -2 if 1280 x 1024 -3 if 1024 x 768 -4 if 640 x 480
-2	16×32 mono-spaced raster
-3	12×24 mono-spaced raster
-4	8×16 mono-spaced raster
-5	16×8 stroke text

Description

This function selects the specified font to be the active font. If you select a font_id that is not defined the command is ignored.

Select a Font — LSSFONT

Return Value None.

Error None.

See Also lsdfont, lsrfont

LSSGC Select Graphics Cursor Definition

Format

- LIB Shell: 0x0080 cursor_id
- C Binding: void lssgc(cursor_id);
- TMLS Interpreter: lssgc cursor_id

Parameters

- short cursor_id: (Default = -1) the label of a graphics cursor definition. If this cursor is not defined then the command is ignored. Two pre-defined graphics cursors are provided:

cursor_id:	graphics cursor shape:
-1	arrow cursor
-2	cross hair cursor

If the cursor selected is not defined then this command is ignored. On power up cursor definition -1 is selected.

Description

This function selects the specified graphics cursor definition for subsequent cursor operations.

Return Value None.

Error None.

See Also krgcsdm, lsrgc, lssgc, krgcen, krgcdi, krgcsda

Skip Words — LSSKIP

Format

- LIB Shell: 0x0120 num_words
- C Binding: void lsskip(num_words);
- TMLS Interpreter: lsskip num_words

Parameters

- unsigned short num_words: the number of words to skip when reading a FIFO or executing a command list.

Description

This command allows you to skip words when executing a command list or when writing data to the Write FIFO.

If the LIB Shell encounters an lsskip command while executing a command list it skips the next num_words of command list data. The lsskip command must be inserted on a command boundary. This is useful for command list modification when the new command list is smaller than the old one.

When you send an lsskip command to the Write FIFO the LIB Shell discards the next num_words of data you transmit to the Write FIFO.

Return Value None.

Error None.

See Also lsclmodify

LSSWLAPAT Select Wide Line Fill Pattern

Format

- LIB Shell: 0x003B apat_id
- C Binding: void lsswlapat(apat_id);
- TMLS Interpreter: lsswlapat apat_id

Parameters

- short apat_id: (Default = -1) the label of a fill pattern definition. Four pre-defined fill patterns are provided:

apat_id:	fill pattern:
-1	solid
-2	hollow
-3	brick
-4	diagonal lines

If the fill pattern selected is not defined then this command is ignored. On power up the fill pattern -1 is selected.

Description

This function selects the fill pattern to be used for subsequent wide line fills. If the apat_id parameter is not equal to -1, then pixel-wide lines will be drawn using the selected pattern as well. In order to use pixel-wide line patterns, set apat_id to -1.

Return Value None.

Error None.

See Also lsdapat, lsrapat, krhwsils, scolsw, scolseshp, scarspin

Set Word Placement Variable LSSWPVAR

Format

- LIB Shell: 0x0076 tot_space no_words
- C Binding: void lsswpvar(tot_space, no_words);
- TMLS Interpreter: lsswpvar tot_space no_words

Parameters

- short tot_space: the total number of pixels to spread between words.
- short no_words: the number of break characters in the string.

Description

This function computes and sets the interword spacing, given the total space available to spread between words and the number of break characters.

Return Value None.

Error None.

See Also sctxsbx, sctxsincs, sctxswpw

LSTGLAY Start Layer Definition

Format

- LIB Shell: 0x0141 layer_id
- C Binding: void lstglay(layer_id);
- TMLS Interpreter: lstglay layer_id

Parameters

- unsigned short layer_id: The 16-bit layer descriptor.

Description

This command marks the beginning of a layer. All following commands will belong to that layer, until either another layer/object definition or an lsclclose command is encountered.

Note

Although this command can be invoked directly from the FIFO, it is intended primarily for command lists.

Return Value None

Error None

See Also lstgogj, lsclclose

Start Object Definition — LSTGOBJ

Format

- LIB Shell: 0x0140 obj_id
- C Binding: void lstgobj(obj_id);
- TMLS Interpreter: lstgobj obj_id

Parameters

- unsigned long obj_id: The 32-bit object descriptor.

Description

This command marks the beginning of an object. All following commands will belong to that object, until either another object definition, or an lsclclose command is encountered.

Note

Although this command can be invoked directly from the FIFO, it is intended primarily for command lists.

Return Value None

Error None

See Also lstglay, lsclclose

LSTGRLAY Delete a Layer

Format

- LIB Shell: 0x143 cl_id obj_id
- C Binding: void lstgrlay(cl_id, obj_id);
- TMLS Interpreter: lstgrlay cl_id obj_id

Parameters

- unsigned short cl_id: The 16-bit command list descriptor.
- unsigned long obj_id: The 32-bit object descriptor.

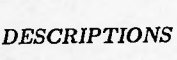
Description

This command deletes a layer from a command list. In reality, the object definition is replaced by an internal command which causes the object to be skipped when the command list is executed. Use the lsclcompr command to reclaim the memory made available by the deleted layer.

Return Value None

Error None

See Also lstgrlay, lsclcompr

Delete an Object — LSTGROBJ 

Format

- LIB Shell: 0x0142 cl_id obj_id
- C Binding: void lstgobj(cl_id, obj_id);
- TMLS Interpreter: lstgobj cl_id obj_id

Parameters

- unsigned short cl_id: The 16-bit command list descriptor.
- unsigned long obj_id: The 32-bit object descriptor.

Description

This command deletes an object from a command list. The object definition is actually replaced by an internal command which causes it to be skipped when the command list is executed. To reclaim the memory made available, use the lsclcompr command.

Return Value None

Error None

See Also lstgrlay, lsclcompr

LSTGSLVIS Set Layer Visibility

Format

- LIB Shell: 0x0144 layer_id vis_flag
- C Binding: void lstgslvis(layer_id, vis_flag):
- TMLS Interpreter: lstgslvis layer_id vis_flag

Parameters

- unsigned short layer_id: The 16-bit layer descriptor.
- unsigned short vis_flag: The new layer visibility flag.

Description

This command enables (if flag = 1), or disables (if flag=0) the current visibility of a layer. When visibility is disabled, all commands in layers with the specified ID will be skipped. It is, however, only possible to set the visibility for layers 0 to 255. All layers with higher IDs will always be visible. The command will take effect at the next lstglay command. Any current layer is unaffected by the command.

Note

To set the visibility of all layers globally, use the lstgsvtbl command.

Return Value None

Error None

See Also lstglay, lstgsvtbl

Set Layer Visibility Table LSTGSVTBL

Format

- LIB Shell: 0x0145 word_0 word_1 ... word_14 word_15
- C Binding: void lstgsvtbl(vtable);
- TMLS Interpreter: lstgsvtbl word_0 word_1 ... word_14 word_15

Parameters

- unsigned short word_0, ... ,word_15: Bit-encoded visibility data.
- unsigned short vtable[]: Array of bit-encoded visibility data.

Description

This command sets the visibility for all layers. The parameters are bit-encoded, as follows:

- bits 0 to 15 of the first word affect layers 0 to 15

.
.
.

- bits 0 to 15 of the last word affect layers 240 to 255

It is, however, only possible to set the visibility for layers 0 to 255. All layers with higher IDs will always be visible. The command will take effect at the next lstglay, that is, any current layer is unaffected.

Return Value None

Error None

See Also lstglay, lstgslvis

LSVSINT Interrupt Host On VSYNC

Format

- LIB Shell: 0x0135 interval
- C Binding: void lsvsint(interval);
- TMLS Interpreter: lsvsint interval

Parameters

- short interval: the interval on which to interrupt the host, measured in frame times (1 frame time = $\frac{1}{60}$ second)

Description

This function is used to interrupt the host at specified intervals for synchronization purposes. For example, if you set interval = 1 then the LIB Shell will interrupt the host on every VSYNC interrupt.

Note

Interrupts generated by the lsvsint command use interrupt line IRQ7. When the host is interrupted by this command, the INTOUT bit of the HSTCTL register is set to one, and a MSGOUT value of 0x011B is reported. This error condition can be cleared by the host, simply by clearing the INTOUT bit.

Return Value None.

Error None.

See Also lsnoop

Wait LSWAIT

Format

- LIB Shell: 0x0121 frames
- C Binding: void lswait(frames);
- TMLS Interpreter: lswait frames

Parameters

- unsigned short frames: the time to wait expressed in terms of frames. A frame time is $\frac{1}{60}$ of a second. The maximum value of frames is 65536; this would produce a wait time of 18 minutes.

Description

This function suspends command execution for the specified length of time.

Return Value None.

Error None.

See Also lsvsint

LSWRRST — Write Raster Image

Format

- LIB Shell: 0x001F left top right bottom data
- C Binding: void lswrrst(left, top, right, bottom, src_filename);
- TMLS Interpreter: lswrrst left top right bottom data

Parameters

- short left: the x-coordinate of the top left corner of the screen rectangle.
- short top: the y-coordinate of the top left corner of the screen rectangle.
- short right: the x-coordinate of the bottom right corner of the screen rectangle.
- short bottom: the y-coordinate of the bottom right corner of the screen rectangle.
- short data[]: the buffer containing the raster data. The color indices are stored as:

4 bits/pixel:

$Pixel_{i+3}$	$Pixel_{i+2}$	$Pixel_{i+1}$	$Pixel_i$

15 0

8 bits/pixel:

$Pixel_{i+1}$	$Pixel_i$

15 0

- char src_filename[]: binary file where the raster image is read from.

Write Raster Image LSWRRST

Description

This function displays an image on the screen from a series of color index data fetched from buffer data. It fills the rectangle from top to bottom, left to right. The image is bounded by the specified screen rectangle.

Return Value None.

Error None.

See Also lsrdrst, lsrlrdrst, lsrlwrrst

SCARAPLYGN Alternate-Fill Polygon

Format

- LIB Shell: 0x0051 nvert $vert_1$ . . . $vert_{nvert}$
- C Binding: void scaraplygn(nvert, vertptr);
- TMLS Interpreter: scaraplygn nvert $vert_1$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of polygon vertices.
- PHCOORD $vert_i$: the coordinates of the i^{th} vertex.
- PHCOORD *vertptr: a pointer to the buffer containing the polygon vertices.

Description

This function fills a polygon with the current area fill pattern, according to the alternate-fill algorithm. It closes the polygon on itself before starting the fill operation.

Return Value None.

Error None.

See Also scarwplygn, lssafapat

Boundary Seed Fill — SCARBFILL

Format

- LIB Shell: 0x0057 x y color_index
- C Binding: void scarbfill(x, y, color_index);
- TMLS Interpreter: scarbfill x y color_index

Parameters

- short x: the x-coordinate of the seed position.
- short y: the y-coordinate of the seed position.
- unsigned short color_index: the boundary fill color index.

Description

This function performs a boundary-type seed fill around the seed location (x, y), up to the boundary defined by color_index, in the current foreground and background colors.

Return Value None.

Error MEMORY_ERROR

See Also lssafapat, scarffill

SCARELLIPSE — Draw a Filled Ellipse

Format

- LIB Shell: 0x0053 a b cx cy
- C Binding: void scarellipse(a, b, cx, cy);
- TMLS Interpreter: scarellipse a b cx cy

Parameters

- unsigned short a: the horizontal radius of the ellipse.
- unsigned short b: the vertical radius of the ellipse.
- short cx: the x-coordinate of the center of the ellipse.
- short cy: the y-coordinate of the center of the ellipse.

Description

This function fills an ellipse in the current area pattern, centered at (cx, cy) with radii a and b.

Return Value None.

Error None.

See Also lssafapat, scareseg, scarepie

Draw a Filled Elliptic Pie Section SCAREPIE

Format

- LIB Shell: 0x0054 sth eth a b cx cy
- C Binding: void scarepie(sth, eth, a, b, cx, cy);
- TMLS Interpreter: scarepie sth eth a b cx cy

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- unsigned short a: the horizontal radius of the ellipse.
- unsigned short b: the vertical radius of the ellipse.
- short cx: the x-coordinate of the center of the ellipse.
- short cy: the y-coordinate of the center of the ellipse.

Description

This function fills an elliptic pie section in the current area pattern. The pie section is based on an ellipse centered at (cx, cy) with radii a and b. The pie section is defined by the start angle sth and the end angle eth.

Return Value None.

Error None.

See Also Issafapat, scarellipse, scareseg

SCARESEG Draw a Filled Elliptic Segment

Format

- LIB Shell: 0x0055 sth eth cx cy a b
- C Binding: void scareseg(sth, eth, a, b, cx, cy);
- TMLS Interpreter: scareseg c a b sth eth

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- unsigned short a: the horizontal radius of the ellipse.
- unsigned short b: the vertical radius of the ellipse.
- short cx: the x-coordinate of the center of the ellipse.
- short cy: the y-coordinate of the center of the ellipse.

Description

This function fills an elliptic segment in the current area pattern. The segment is based on an ellipse centered at (cx, cy) with radii a and b. The segment is defined by the start angle sth and and the end angle eth.

Return Value None.

Error None.

See Also Issafapat, scarellipse, scarepie

Flood Seed Fill — SCARFFILL

Format

- LIB Shell: 0x0056 x y
- C Binding: void scarffill(x, y);
- TMLS Interpreter: scarffill x y

Parameters

- short x: the x-coordinate of the seed position.
- short y: the y-coordinate of the seed position.

Description

This function performs a seed fill around the seed location (x, y), while it encounters pixels having the same color index as the seed position. The fill is performed using the current foreground and background colors.

Return Value None.

Error None.

See Also krhwsbcol, krhwsfcol, lssafapat, scarbfill

SCARRECT Draw a Filled Rectangle

Format

- LIB Shell: 0x004F x_0 y_0 x_1 y_1
- C Binding: void scarrect(x_0, y_0, x_1, y_1);
- TMLS Interpreter: scarrect x_0 y_0 x_1 y_1

Parameters

- short x_0: the x-coordinate of one corner of the rectangle.
- short y_0: the y-coordinate of one corner of the rectangle.
- short x_1: the x-coordinate of the opposing corner of the rectangle.
- short y_1: the y-coordinate of the opposing corner of the rectangle.

Description

This function fills a rectangle with opposing corners at (x_0, y_0) and (x_1, y_1).

Return Value None.

Error None.

See Also lssafapat, scarrrect, scartri

Filled Rounded Rectangle SCARRRECT

Format

- LIB Shell: 0x0050 x_0 y_0 x_1 y_1 a b
- C Binding: void scarrrect(x_0, y_0, x_1, y_1, a, b);
- TMLS INTERPRETER: scarrrect X_0 y_0 x_1 y_1 a b

Parameters

- short x_0: the x-coordinate of one corner of the rectangle.
- short y_0: the y-coordinate of one corner of the rectangle.
- short x_1: the x-coordinate of the opposing corner of the rectangle.
- short y_1: the y-coordinate of the opposing corner of the rectangle.
- unsigned short a: the horizontal radius used to round off the corner.
- unsigned short b: the vertical radius used to round off the corner.

Description

This function fills a rounded rectangle with opposing corners at (x_0, y_0) and (x_1, y_1). The radii a and b are used to round off the corners of the rectangle.

Return Value None.

Error None.

See Also Issapat, scarrect, scartri

SCARSPIN — Set Pattern Pinning

Format

- LIB Shell: 0x005C pin_flag
- C Binding: void scarspin(pin_flag);
- TMLS Interpreter: scarspin pin_flag

Parameters

- short pin_flag: (Default = 0) the pattern pinning flag. The following values are expected:
  - null: disable pattern pinning. Subsequent pattern fills will use the top left corner of the screen as the pattern pin point.
  - non-null: enable pattern pinning. Subsequent pattern fills will use the current screen origin as the pattern pin point.

Description

This function is used to enable or disable pinning fill patterns to the current screen origin.

Normally, when the shell fills an object it "pins" the top left corner of the fill pattern to the top left corner of the screen and replicates the pattern across and down the screen to fill the area in question. Alternately, you can use this command to pin the top left corner of the fill pattern to the current screen origin.

Return Value None.

Error None.

See Also scsorg, lsafapat, lswlapat

Draw a Filled Triangle — SCARTRI

Format

- LIB Shell: 0x005B x_1 y_1 x_2 y_2 x_3 y_3
- C Binding: void scartri (x_1, y_1, x_2, y_2, x_3, y_3);
- TMLS Interpreter: scartri x_1 y_1 x_2 y_2 x_3 y_3

Parameters

- short x_i: the x-coordinate of the i^{th} corner of the triangle.
- short y_i: the y-coordinate of the i^{th} corner of the triangle.

Description

This function draws a filled triangle.

Return Value None.

Error None.

See Also lssafapat, scarrect, scarrrect

SCARWPLYGN Winding-Number Fill Poly.

Format

- LIB Shell: 0x0052 nvert $vert_1$ $vert_2$ ... $vert_{nvert}$
- C Binding: void scarwplygn(nvert, vertptr);
- TMLS Interpreter: scarwplygn nvert $vert_1$ $vert_2$ ... $vert_{nvert}$

Parameters

- unsigned short nvert: the number of polygon vertices.
- PHCOORD $vert_i$: the coordinates of the i^{th} vertex.
- PHCOORD *vertptr: a pointer to the buffer containing the polygon vertices.

Description

This function fills the specified polygon, in the current area fill pattern, using the winding-number fill algorithm. It closes the specified polygon on itself before starting the fill operation.

Return Value None.

Error None.

See Also lssafapat, scaraplygn

Draw an Elliptic Arc — SCOLEARC

Format

- LIB Shell: 0x0040 sth eth a b cx cy
- C Binding: void scolearc(sth, eth, a, b, cx, cy);
- TMLS Interpreter: scolearc sth eth a b cx cy

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- unsigned short a: the horizontal radius of the elliptic arc.
- unsigned short b: the vertical radius of the elliptic arc.
- short cx: the x-coordinate of the center of the arc.
- short cy: the y-coordinate of the center of the arc.

Description

This function draws an elliptic arc, based on an ellipse centered at (cx, cy) with radii a and b. The arc is defined by the start angle sth and the end angle eth.

Return Value None.

Error None.

See Also scolellipse, scoleseg, scolepie, scolhpearc

SCOLELLIPSE Draw an Ellipse Outline

Format

- LIB Shell: 0x003F a b cx cy
- C Binding: void scolellipse(a, b, cx, cy);
- TMLS Interpreter: scolellipse a b cx cy

Parameters

- unsigned short a: the horizontal radius of the ellipse.
- unsigned short b: the vertical radius of the ellipse.
- short cx, cy: the center of the ellipse.

Description

This function draws an ellipse centered at (cx, cy) with radii a and b.

Return Value None.

Error None.

See Also scoleseg, scolepie, scolearc, scolhpearc

Draw an Elliptic Pie Outline SCOLEPIE

Format

- LIB Shell: 0x0041 sth eth a b cx cy
- C Binding: void scolepie(sth, eth, a, b, cx, cy);
- TMLS Interpreter: scolepie sth eth a b cx cy

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- unsigned short a: the horizontal radius of the elliptic pie section.
- unsigned short b: the vertical radius of the elliptic pie section.
- short cx: the x-coordinate of the center of the pie section.
- short cy: the y-coordinate of the center of the pie section.

Description

This function draws an elliptic pie section based, on an ellipse centered at (cx, cy) with radii a and b. The pie section is defined by the start angle sth and and the end angle eth.

Return Value None.

Error None.

See Also scolellipse, scoleseg, scolearc, scolhpearc

SCOLESEG Elliptic Segment Outline

Format

- LIB Shell: 0x0042 sth eth a b cx cy
- C Binding: void scoleseg(sth, eth, a, b, cx, cy);
- TMLS Interpreter: scoleseg sth eth a b cx cy

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- unsigned short a: the horizontal radius of the elliptic segment.
- unsigned short b: the vertical radius of the elliptic segment.
- short cx: the x-coordinate of the center of the segment.
- short cy: the y-coordinate of the center of the segment.

Description

This function draws an elliptic segment, based on an ellipse centered at (cx, cy) with radii a and b. The segment is defined by the start angle sth and and the end angle eth.

Return Value None.

Error None.

See Also scolellipse, scoleseg, scolearc, scolhpearc

High Prec. Elliptic Arc — SCOLHPEARC

Format

- LIB Shell: 0x0047 sth eth a b cx cy
- C Binding: void scolhpearc(sth, eth, a, b, cx, cy);
- TMLS Interpreter: scolhpearc sth eth a b cx cy

Parameters

- FIXPT sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- FIXPT eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- unsigned short a: the horizontal radius of the elliptic arc.
- unsigned short b: the vertical radius of the elliptic arc.
- short cx: the x-coordinate of the center of the arc.
- short cy: the y-coordinate of the center of the arc.

Description

This function draws a high precision elliptic arc, based on an ellipse centered at (cx, cy) with radii a and b. The arc is defined by the start angle sth and the end angle eth.

Return Value None.

Error None.

See Also scolellipse, scoleseg, scolepie

SCOLHSPLINE Pseudo-Hermite Spline

Format

- LIB Shell: 0x0043 x_0 y_0 x_1 y_1 x_2 y_2 x_3 y_3 nsteps
- C Binding: void scolhspline(x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3, nsteps);
- TMLS Interpreter: scolhspline x_0 y_0 x_1 y_1 x_2 y_2 x_3 y_3 nsteps

Parameters

- short x_i: the x-coordinate of the i^{th} control point.
- short y_i: the y-coordinate of the i^{th} control point.
- unsigned short nsteps: the number of line segments in each spline section.

Description

This function draws a Hermite-type spline by joining even control points in the direction of odd control points. Thus, the spline starts at (x_0, y_0) in the direction of (x_1, y_1), then goes in the direction of (x_3, y_3) to join (x_2, y_2). The nsteps parameter specifies the number of line segments used to join even control points (x_0, y_0) and (x_2, y_2).

Return Value None.

Error OVERFLOW_ERROR

See Also scolpline, scolplygn

Draw a Line — SCOLLNTO

Format

- LIB Shell: 0x0026 x y
- C Binding: void scollnto(x, y);
- TMLS Interpreter: scollnto x y

Parameters

- short x: the x-coordinate of a screen position.
- short y: the y-coordinate of a screen position.

Description

This function draws a line starting at the current lineto-moveto position and ending at the specified position. The lineto-moveto position is updated to the specified end position.

Return Value None.

Error None.

See Also scolmvto, krhwsils, scolsmode

SCOLMVTO Set Lineto-Moveto Position

Format

- LIB Shell: 0x0025 x y
- C Binding: void scolmvto(x, y);
- TMLS Interpreter: scolmvto x y

Parameters

- short x: (Default = 0) the x-coordinate of the start position.
- short y: (Default = 0) the y-coordinate of the start position.

Description

This function sets the lineto-moveto position from which subsequent line drawings will start.

Return Value None.

Error None.

See Also scollnto

Poly-Line SCOLPLINE

Format

- LIB Shell: 0x003C nvert $vert_1$ . . . $vert_{nvert}$
- C Binding: void scolpline(nvert, vertptr);
- TMLS Interpreter: scolpline nvert $vert_1$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of poly-line vertices.
- PHCOORD $vert_i$: the i^{th} poly-line vertex.
- PHCOORD *vert: a pointer to the buffer containing the poly-line vertices.

Description

This function joins a series of vertices with line segments. Vertices are connected in the order they are specified.

Return Value None.

Error None.

See Also scolplygn, scolhspline

SCOLPLYGN — Polygon

Format

- LIB Shell: 0x0045 nvert $vert_1$ . . . $vert_{nvert}$
- C Binding: void scolplygn(nvert, vertptr);
- TMLS Interpreter: scolplygn nvert $vert_1$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of poly-line vertices.
- PHCOORD $vert_i$: the i^{th} poly-line vertex.
- PHCOORD *vert: a pointer to the buffer containing the poly-line vertices.

Description

This function draw a polygon by connecting the vertices in the order they are specified. The last vertex is joined with the first to close the polygon.

Return Value None.

Error None.

See Also scolpline, scolhspline

Draw a Rectangle Outline SCOLRECT

Format

- LIB Shell: 0x003D x_0 y_0 x_1 y_1
- C Binding: void scolrect(x_0, y_0, x_1, y_1);
- TMLS Interpreter: scolrect x_0 y_0 x_1 y_1

Parameters

- short x_0: the x-coordinate of one corner of the rectangle.
- short y_0: the y-coordinate of one corner of the rectangle.
- short x_1: the x-coordinate of the opposing corner of the rectangle.
- short y_1: the y-coordinate of the opposing corner of the rectangle.

Description

This function draws a rectanglar outline with opposing corners at (x_0, y_0) and (x_1, y_1).

Return Value None.

Error None.

See Also scolrrect

SCOLRRECT Rounded Rectangle Outline

Format

- LIB Shell: 0x003E x_0 y_0 x_1 y_1 a b
- C Binding: void scolrrect(x_0, y_0, x_1, y_1, a, b);
- TMLS Interpreter: scolrrect x_0 y_0 x_1 y_1 a b

Parameters

- short x_0: the x-coordinate of one corner of the rounded rectangle.
- short y_0: the y-coordinate of one corner of the rounded rectangle.
- short x_1: the x-coordinate of the opposing corner of the rounded rectangle.
- short y_1: the y-coordinate of the opposing corner of the rounded rectangle.
- unsigned short a: the horizontal radius used to round off the corners.
- unsigned short b: the vertical radius used to round off the corners.

Description

This function draws the outline of a rounded rectangle with opposing corners at (x_0, y_0) and (x_1, y_1). The radii a and b are used to round off the corners of the rectangle.

Return Value None.

Error None.

See Also scolrect

Set Endpoint Shape SCOLSESHP

Format

- LIB Shell: 0x003A begin_sh middle_sh end_sh
- C Binding: void scolseshp(begin_sh, middle_sh, end_sh);
- TMLS Interpreter: scolseshp begin_sh middle_sh end_sh

Parameters

- short begin_sh: (Default = 0) the endpoint shape for the first outline point:
  - 0: none
  - 1: round
  - 2: square
  - 3: arrow1
- short middle_sh: (Default = 3) the shape that joins two segments:
  - *0:* none
  - *1:* round
  - *2:* miter
  - *3:* bevel
- short end_sh: (Default = 0) the endpoint shape for the last outline point: same as for begin_sh.

Description

This function sets the shape to be drawn at the endpoints of open outlines.

Pixel-wide outlines' endpoints either end in a dot or an arrowhead. In this case endpoint shape options 0 and 1 in effect mean "do not draw endpoint shapes".

SCOLSESHP Set Endpoint Shape

Round Miter Bevel

Note Endpoint shapes are only applied to poly-lines and elliptical outlines that are drawn in screen or 2D coordinate space. The scolseshp command must be followed by a v2olsw (2D Outline Width) command in order for the new start, joint, and end shapes to be reflected in the virtual 2D layer.

Return Value None.

Error None.

See Also krhwsils, scolsw, lsswlapat

Set Outline Mode SCOLSMODE

Format

- LIB Shell: 0x0046 mode
- C Binding: void scolsmode(mode);
- TMLS Interpreter: scolsmode mode

Parameters

- unsigned short mode: the outline mode. The low byte is the continuity flag (Default = null):
  - null: disable line style continuity. The shell resets the line style to the initial line style before each outline primitive is drawn.
  - non-null: enable line style continuity. When you draw a series of outline primitives the line style continues from one outline to the next.

The high byte is the endpoint flag (Default = non-null):

null: the last point of each pixel-wide outline is not drawn.

non-null: the last point of each pixel-wide outline is drawn.

Description

This function sets the outline mode.

Return Value None.

Error None.

See Also krhwgcls, krhwscls, krhwrcls, krhwsils

SCOLSW — Set Outline Width

Format

- LIB Shell: 0x0039 new_width
- C Binding: void scolsw(new_width);
- TMLS Interpreter: scolsw new_width

Parameters

- short new_width: (Default = 1) the outline width. A negative or null width indicates a null pen, meaning that no outline will be generated. A width of 1 pixel indicates standard 1-pixel wide outlines. A width greater than one indicates wide-line drawing.

Description

This function sets the new line width. The width is internally forced to be odd by rounding up even width specifications.

Return Value None.

Error None.

See Also scolseshp, krhwsils, lsswlapat

Set Screen Origin — SCSORG

Format

- LIB Shell: 0x0048 x y
- C Binding: void scsorg (x, y);
- TMLS Interpreter: scsorg x y

Parameters

- unsigned short x: (Default = 0) the x-coordinate of the new screen origin with respect to the top left corner of the screen.
- unsigned short y: (Default = 0) the y-coordinate of the new screen origin with respect to the top left corner of the screen.

Description

This function allows you to reset the screen origin for subsequent screen-based commands.

Return Value None.

Error None.

See Also scarspin

SCSTHINCR Set Theta Increment

Format

- LIB Shell: 0x0044 theta
- C Binding: void scsthincr(theta);
- TMLS Interpreter: scsthincr theta

Parameters

- ANGLE theta: (Default = 5°) the increment angle for drawing certain types of curves. It only applies to:
  - outlines drawn in screen coordinates having line widths greater than one.
  - filled primitives with one or more curved edges.
  - large elliptic outlines.
  - large elliptic arcs.

Description

This function is used to specify the precision of certain types of curve drawing. Most curves are rendered using differential data algorithms. However, for the primitives mentioned above we use trigonometric curve emulation. The angle specified by this command defines the precison of the emulation process. It is the angular increment at which points on the curve are calculated and joined. Thus, the smaller the value of theta the more precise the curve will be.

Note that rendering precison is inversely proportional to the rendering speed.

Set Theta Increment — SCSTHINCR

Return Value None.

Error None.

See Also lsswlapat, scolsw

SCTXBEXTENT Determine String Extent

Format

- LIB Shell: (not available)
- C binding: void sctxbextent(x, y, strptr);
- TMLS Interpreter: (not available)

Parameters

- short x, y: the destination coordinates on the screen.
- char *strptr: a pointer to the character string.

Description

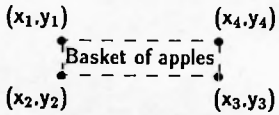
This function calculates the four boundary points of the specified string if you were to draw it on the screen. **This command does not draw the string**; use the sctxstring, the v2txstring, or the sctxbstring command to draw strings on the screen.

In contrast to the sctxextent command, sctxbextent expects strings of byte-valued characters. sctxbextent is only supported in the C Binding.

Note The sctxbextent command is now redundant but is maintained for compatibility with previous software releases.

Return Value The four coordinates (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) of the string are returned to the host in the Read FIFO (in that order). The drawing below illustrates the location of the coordinates with respect to the string "Basket of apples".

Determine String Extent SCTXBEXTENT



The host retrieves the data with the C Binding's iogetbuf function.

Error None.

See Also sctxbstring, sctxstring, v2txstring, sctxsalign

SCTXBSTRING — Draw Text

Format

- LIB Shell: (not available)
- C Binding: void sctxbstring(x, y, strptr);
- TMLS Interpreter: (not available)

Parameters

- short x: the x-coordinate of the destination point on the screen.
- short y: the y-coordinate of the destination point on the screen.
- char *strptr: a pointer to the character string.

Description

This function draws a string of characters on the screen. It is only supported in the C binding. In contrast to the sctxstring and the v2txstring functions, sctxbstring expects strings of byte-valued characters.

Return Value None.

Error None.

See Also sctxbextent, sctxextent, sctxstring, v2txstring

Determine String Extent SCTXEXTENT

Format

- LIB Shell: 0x0074 x y length string
- C Binding: void sctxextent(x, y, strptr);
- TMLS Interpreter: sctxextent x y "string"

Parameters

- short x, y: the destination coordinates on the screen.
- unsigned short length: the length of the string in characters.
- short string[]: the character string. For the TMLS interpreter the string is simply enclosed in double quotes. For the LIB Shell send the hexadecimal **character codes** of the characters in the string.
- short *strptr: a pointer to the character string.

Description

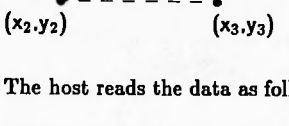
This function calculates the four boundary points of the specified string if you were to draw it on the screen. **This command does not draw the string**; use the sctxstring command to draw strings on the screen.

Note that the C Binding has a function called sctxbextent which supports byte-valued characters.

Note The sctxextent command is now redundant but is maintained for compatibility with previous software releases.

SCTXEXTENT Determine String Extent

Return Value The four coordinates (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) of the string are returned to the host in the Read FIFO (in that order). The drawing below illustrates the location of the coordinates with respect to the string "Basket of apples".



The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the C binding's iogetbuf function to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also sctxbstring, sctxstring, sctxsalign

Set String Alignment Options SCTXSALIGN

Format

- LIB Shell: 0x006B halign valign
- C Binding: void sctxsalign(halign, valign);
- TMLS Interpreter: sctxsalign halign valign

Parameters

- short halign: (Default = 0) the horizontal string alignment code; the following values are expected:
  - 0: left justified
  - 1: right justified
  - 2: centered
- short valign: (Default = 0) the vertical alignment code; the following values are expected:
  - *0:* top
  - *1:* ascent
  - *2:* half
  - *3:* descent
  - *4:* baseline
  - *5:* bottom

Description

This function sets the string alignment options for subsequent text drawing operations.

Note

The horizontal alignment is based on the string length as drawn. The vertical alignment is based on internal font specification data.

SCTXSALIGN Set String Alignment Options

Return Value None.

Error None.

See Also

Set Bolding Factor SCTXSBF

Format

- LIB Shell: 0x0070 bolding_factor
- C Binding: void sctxsbf(bolding_factor);
- TMLS Interpreter: sctxsbf bolding_factor

Parameters

- short bolding_factor: (Default = 0) the bolding factor. The character cell row is replicated one pixel to the left the number of times specified each drawing cycle. A null or negative bolding factor effectively disables bolding. For a bolding factor of 1:

```
---XX---          --XXX---
--X--X--          -XX-XX--
--X--X--   -->    -XX-XX--
--XXXX--          -XXXXX--
--X--X--          -XX-XX--
--X--X--          -XX-XX--
```

Description

This function sets the bolding factor for subsequent drawing operations.

Return Value None.

Error None.

See Also sctxslm, sctxsim

SCTXSBLROT Set Baseline Rotation Angle

Format

- LIB Shell: 0x0069 baseline_angle
- C Binding: void sctxsblrot(baseline_angle);
- TMLS Interpreter: sctxsblrot baseline_angle

Parameters

- ANGLE baseline_angle: (Default = 0°) the angle to use for the string baseline.

Description

This function sets the string baseline rotation angle for subsequent text drawing operations. The value of the baseline rotation angle affects which character rotation angles are supported. Refer to the description of the sctxschrot command for more details.

Return Value None.

Error None.

See Also sctxschrot

Set Extra Space SCTXSBX

Format

- LIB Shell: 0x0075 extra_space
- C Binding: void sctxsbx(extra_space);
- TMLS Interpreter: sctxsbx extra_space

Parameters

- FIXPT extra_space: (Default = 0.0) the extra space between two consecutive words (i.e. interword spacing) in a text string.

Description

This function allows the user to set the amount of extra space to spread between words in strings. The amount of extra space can also be computed by the word placement algorithm (refer to the sctxswpw command description). It is usually the difference between the string extent and the desired string length, divided by the number of break characters in the string.

Return Value None.

Error None.

See Also sctxsincs, sctxswpw, lsswpvar

SCTXSCHROT Char. Cell Rotation Angle

Format

- LIB Shell: 0x006A rot_angle
- C Binding: void sctxschrot(rot_angle);
- TMLS Interpreter: sctxschrot rot_angle

Parameters

- ANGLE rot_angle: (Default = 0°) the character cell rotation angle. The values allowed depend on the current value of the baseline rotation angle and the type of text being drawn. For stroke text the character rotation angle is either 0° or the current baseline rotation angle. For raster text the following character rotation angles are possible:

Baseline Rotation Angle	Character Rotation Angle
0°	0°
90°	0° or 90°
180°	0° or 180°
270°	0° or 270°
360°	0°

Description

This function set the character cell rotation angle for subsequent text drawing operations.

Return Value None.

Error None.

See Also sctxsblrot

Set String Drawing Mode SCTXSDMODE

Format

- LIB Shell: 0x0079 mode
- C Binding: void sctxsdmode(mode);
- TMLS Interpreter: sctxsdmode mode

Parameters

- short mode: (Default = 0) The string drawing mode. The three least significant bits control the output of sctxstring, as well as its return values.
  - Bit 0: If one, sctxstring and v2txstring do not draw any text. Otherwise, text is drawn normally.
  - Bit 1: If one, sctxstring and v2txstring report the top left corner coordinate of each character to the Read FIFO.
  - Bit 2: If one, sctxstring and v2txstring report the string extent to the Read FIFO.
  - Bits 3-15: Not used.

Description

This command controls the output and return values of the sctxstring and the v2txstring commands. See the bit descriptions above. All eight combinations are possible.

The sctxextent and sctxbextent commands are now redundant but are maintained for compatibility with previous software releases.

SCTXSDMODE Set String Drawing Mode

Return Value None. However, the selected drawing mode could modify sctxstring's return values.

Error None.

See Also sctxstring, sctxextent

Set Italicization Mask — SCTXSIM

Format

- LIB Shell: 0x006E ital_mask
- C Binding: sctxsim(ital_mask);
- TMLS Interpreter: sctxsim ital_mask

Parameters

- unsigned short ital_mask: (Default = 0) the italicization mask. A 1 in the Most Significant Bit (MSB) of the mask indicates that a horizontal offset by one pixel should occur. A 0 in the MSB of the mask means the offset used for the previous line should be used. The mask is rotated left after each character cell row. Some typical values are:

  0x0000: no italicization.

  0x5555: standard 63° italic.

  0xFFFF: 45° italic.

  The following example shows how the letter "A" is italicized with a mask of 0x5555:

```
---XX---          ------XX
--X--X--          ----X--X
--X--X--   -->    ----X--X
--XXXX--          ---XXXX-
--X--X--          ---X--X-
--X--X--          --X--X--
```

Description

This function sets the italicization mask for subsequent text drawing operations.

SCTXSIM — Set Italicization Mask

Return Value None.

Error None.

See Also sctxsbf, sctxslm

Set Intercharacter Spacing SCTXSINCS

Format

- LIB Shell: 0x0073 inter_char
- C Binding: void sctxsincs(inter_char);
- TMLS Interpreter: sctxsincs inter_char

Parameters

- short inter_char: (Default = 0) the intercharacter spacing value.

Description

This function sets the amount of space between characters.

Return Value None.

Error None.

See Also sctxsbx, sctxswpw, lsswpvar

SCTXSLM Set Lightening Mask

Format

- LIB Shell: 0x006F lightmask
- C Binding: void sctxslm(lightmask);
- TMLS Interpreter: sctxslm lightmask

Parameters

- unsigned short lightmask: (Default = 0) the lightening mask. Some typical values are:
  - 0x0000: Disable lightening. Full intensity.
  - 0x5555: Standard lightening. Half-intensity.
  - 0xFFFF: Every other character row is blanked.

Description

This function sets the lightening mask for subsequent text drawing. The mask is ANDed with each character cell row, dimming the apparent character intensity. After each row is processed the mask is 1-complemented in order to avoid blank vertical streaks through the text. For example, with light_mask set to 0x5555:

```
---XX---          ----X---
--X--X--          -----X--
--X--X--   -->    --X-----
--XXXX--          ---X-X--
--X--X--          --X-----
--X--X--          -----X--
```

Return Value None.

Error None.

See Also sctxsbf, sctxsim

Set Text Scaling SCTXSSF

Format

- LIB Shell: 0x0068 xscale yscale
- C Binding: void sctxssf(xscale, yscale);
- TMLS Interpreter: sctxssf xscale yscale

Parameters

- FIXPT xscale: (Default = 1.0) the x-axis scaling factor. A positive value scales the text with respect to the x-axis. A negative value performs a mirror transformation of the text as well. A value of zero means the shell does not draw text.
- FIXPT yscale: (Default = 1.0) the y-axis scaling factor. A positive value scales the text with respect to the y-axis. A negative value performs a mirror transformation of the text as well. A value of zero means the shell does not draw text.

Description

This function sets the text scaling factors for subsequent text drawing operations.

Return Value None.

Error None.

See Also

SCTXSSOW Set Strike-Out Thickness

Format

- LIB Shell: 0x006D strike_out_thick
- C Binding: void sctxssow(strike_out_thick);
- TMLS Interpreter: sctxssow strike_out_thick

Parameters

- unsigned short strike_out_thick: (Default = 0) the strike-out thickness. Null or negative strike-out thicknesses disable striking out.

Description

This function sets the strike-out thickness for subsequent text drawing. Striking-out is always performed centered on the half line.

Return Value None.

Error None.

See Also

Set Stroke Width SCTXSSW

Format

- LIB Shell: 0x0077 stroke_width
- C Binding: void sctxssw (stroke_width);
- TMLS Interpreter: sctxssw stroke_width

Parameters

- unsigned short stroke_width: (Default = 1) the stoke width for stroke-based fonts.

Description

This function allows you to set the width of the lines (strokes) in which stroke characters are drawn.

Return Value None.

Error None.

See Also lsdfont

SCTXSTRING — Draw Screen Text

Format

- LIB Shell: 0x0072 x y length string
- C Binding: void sctxstring(x, y, strptr);
- TMLS Interpreter: sctxstring x y "string"

Parameters

- short x: the x-coordinate of the destination point on the screen.
- short y: the y-coordinate of the destination point on the screen.
- short string[]: the character string. For the TMLS interpreter the string is simply enclosed in single or double quotes. For the LIB Shell send the hexadecimal character codes of the characters in the string.
- unsigned short length: the length of the string in characters.
- short *strptr: a pointer to the character string.

Description

This function draws a string of characters on the screen. The function expects strings of word-valued characters. Note that the C Binding has a function called sctxbstring which supports strings of byte-valued characters.

If string drawing mode bit 0 is one (see sctxsdmode), text is not drawn.

Return Value Values returned in the Read FIFO depend on the string drawing mode. If drawing mode bits 1 and 2 are both 0, there will be no return value.

Draw Screen Text — SCTXSTRING

The following table summarizes what is returned for each combination of drawing mode bits 1 and 2. The coordinates refer to the drawing below.

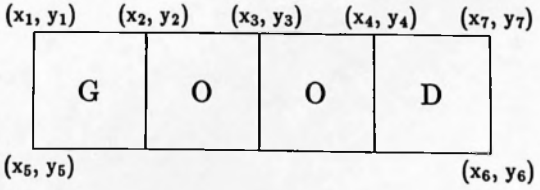
The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the C binding iogetbuf function to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Drawing Mode Bit 2	Drawing Mode Bit 1	Order of coordinates in the Read FIFO.
0	0	None
0	1	(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)
1	0	(x_1, y_1), (x_5, y_5), (x_6, y_6), (x_7, y_7)
1	1	(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)
		(x_1, y_1), (x_5, y_5), (x_6, y_6), (x_7, y_7)



Error None.

See Also sctxsdmode, sctxbextent, sctxbstring, sctxextent, v2txstring

SCTXSUW Set Underline Attributes

Format

- LIB Shell: 0x006C underline_thick
- C Binding: void sctxsuw(underline_thick);
- TMLS Interpreter: sctxsuw underline_thick

Parameters

- short underline_thick: (Default = 0) the underline thickness. It is also the distance from the base line and to the underline. Null or negative underline thicknesses disable underlining.

Description

This function sets the string underline thickness; subsequent text drawn will be underlined until underlining is disabled. Underlining is performed underline_thick scan lines below the base line.

Return Value None.

Error None.

See Also

Set Word Placement Width SCTXSWPW

Format

- LIB Shell: 0x0071 width
- C Binding: void sctxswpw(width);
- TMLS Interpreter: sctxswpw width

Parameters

- short width: (Default = 0) the word placement width:

  > 0: Text is forced to span the specified width by inserting the extra interword spacing between consecutive words. Word placement is enabled. The function calculates the necessary extra interword spacing and resets the current extra interword spacing to this value.

  = 0: Text is drawn as specified. The function disables word placement and extra interword spacing is not used. The value set by the last sctxsbx or lsswpvar command is still valid.

  < 0: Text is drawn as specified. The function disables word placement but does use the current extra interword spacing (as set by the most recent sctxsbx or lsswpvar command executed).

Description

This function enables or disables the word placement feature and sets the word placement width.

Return Value None.

Error None.

See Also sctxsbx, sctxsincs, lsswpvar

V2ARAPLYGN Alternate-Fill Polygon

Format

- LIB Shell: 0x00BA nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v2araplygn(nvert, vertptr);
- TMLS Interpreter:v2araplygn Nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of vertices.
- V2COORD $vert_i$: the coordinates of the i^{th} polygon vertex.
- V2COORD *vertptr: a pointer to the buffer containing the polygon vertices.

Description

This function fills the specified polygon using the alternate-fill algorithm. It closes the polygon on itself before starting the fill operation. This function does not affect the 2D current position.

Return Value None.

Error None.

See Also v2ararplygn, v2arwplygn, v2arwrplygn

Alternate-Fill Rel. Polygon V2ARARPLYGN

Format

- LIB Shell: 0x00BB ndelt $delt_1$ $delt_2$ . . . $delt_{ndelt}$
- C Binding: void v2ararplygn(ndelt, deltptr);
- TMLS Interpreter: v2ararplygn ndelt $delt_1$ $delt_2$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of (Δ x, Δy) values.
- V2COORD $delt_i$: the i^{th} (Δx, Δy) value. The delta values are displacements from the 2D current point.
- V2COORD *delt: a pointer to a buffer containing the delta values.

Description

This function fills the relative polygon using the alternate-fill algorithm. It closes the polygon on itself before starting the fill operation. This function does not affect the 2D current position.

Return Value None.

Error None.

See Also v2araplygn

V2ARBFILL — Boundary Fill

Format

- LIB Shell: 0x00C2 color_index
- C Binding: void v2arbfill(color_index);
- TMLS Interpreter: v2arbfill color_index

Parameters

- unsigned short color_index: the color index of the boundary to be filled to.

Description

This function performs a fill around the current 2D position up to the boundary defined by color_index. This function does not affect the 2D current position.

Return Value None.

Error MEMORY_ERROR

See Also v2arffill

Draw a Filled Ellipse V2ARELLIPSE

Format

- LIB Shell: 0x00BE a b
- C Binding: void v2arellipse(a, b);
- TMLS Interpreter: v2arellipse a b

Parameters

- FIXPT a: the horizontal radius of the ellipse.
- FIXPT b: the vertical radius of the ellipse.

Description

This function fills an ellipse centered at the 2D current position with radii a and b. This function does not affect the 2D current position.

Return Value None.

Error OVERFLOW_ERROR

See Also v2arepie, v2areseg

V2AREPIE — Filled Ellipt. Pie Sec.

Format

- LIB Shell: 0x00BF sth eth a b
- C Binding: void v2arepie(sth, eth, a, b);
- TMLS Interpreter: v2arepie sth eth a b

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- FIXPT a: the horizontal radius.
- FIXPT b: the vertical radius.

Description

This function fills an elliptical pie section, based on an ellipse centered at the 2D current position with radii a and b. The pie section is defined by the start angle sth and and the end angle eth. This function does not affect the 2D current position.

Return Value None.

Error OVERFLOW_ERROR

See Also v2arellipse, v2areseg

Draw a Filled Elliptical Segment V2ARESEG

Format

- LIB Shell: 0x00C0 sth eth a b
- C Binding: void v2areseg(sth, eth, a, b);
- TMLS Interpreter: v2areseg sth eth a b

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- FIXPT a: the horizontal radius.
- FIXPT b: the vertical radius.

Description

This function draws a filled elliptical segment, based on an ellipse centered at the 2D current position with radii a and b. The segment is defined by the start angle sth and and the end angle eth. This function does not affect the 2D current position.

Return Value None.

Error OVERFLOW_ERROR

See Also v2arellipse, v2arepie

V2ARFFILL — Flood Seed Fill

Format

- LIB Shell: 0x00C1
- C Binding: void v2arffill();
- TMLS Interpreter: v2arffill

Parameters None.

Description

This function performs a flood fill around the 2D current position, while it encounters pixels having the same color index as the seed position. The 2D current position is not affected.

Return Value None.

Error None.

See Also v2arbfill

Fill Rectangle V2ARRECT

Format

- LIB Shell: 0x00B8 x y
- C Binding: void v2arrect(x, y);
- TMLS Interpreter: v2arrect x y

Parameters

- FIXPT x: the x-coordinate of one corner of the rectangle.
- FIXPT y: the y-coordinate of one corner of the rectangle.

Description

This function fills the 2D virtual rectangle with opposing corners at the 2D current position and the point (x, y). This function does not affect the 2D current position.

Return Value None.

Error None.

See Also v2arrrect

V2ARRRECT Fill a Relative Rectangle

Format

- LIB Shell: 0x00B9 deltx delty
- C Binding: void v2arrrect(deltx, delty);
- TMLS Interpreter: v2arrrect deltx delty

Parameters

- FIXPT deltx: the Δx value defining the x-coordinate of one corner of the rectangle.
- FIXPT delty: the Δy value defining the y-coordinate of one corner of the rectangle.

Description

This function fills a 2D virtual rectangle with opposing corners at the 2D current point (x_o, y_o) and the point (x_o+deltx, y_o+delty). This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2arrect

Winding-Number Poly. Fill V2ARWPLYGN

Format

- LIB Shell: 0x00BC nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v2arwplygn(nvert, vertptr);
- TMLS Interpreter: v2arwplygn nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of polygon vertices.
- V2COORD $vert_i$: the coordinates of the i^{th} polygon vertex.
- V2COORD *vertptr: a pointer to a buffer containing the vertices.

Description

This function fills the specified polygon using the winding-number fill algorithm. It closes the polygon on itself before starting the fill operation. This function does not affect the 2D current position.

Return Value None.

Error None.

See Also v2araplygn, v2ararplygn, v2arwrplygn

V2ARWRPLYGN Winding Fill Rel. Poly.

Format

- LIB Shell: 0x00BD ndelt $delt_1$ $delt_2$ . . . $delt_{ndelt}$
- C Binding: void v2arwrplygn(ndelt, deltptr);
- TMLS Interpreter: v2arwrplygn ndelt $delt_1$ $delt_2$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of (Δx, Δy) values.
- V2COORD $delt_i$: the i^{th} (Δx, Δy) value. The delta values are displacements from the 2D current point.
- V2COORD *deltptr: a pointer to a buffer containing the delta values.

Description

This function fills the relative polygon using the winding-number fill algorithm. It closes the specified polygon on itself before starting the fill operation. This function does not affect the 2D current position.

Return Value None.

Error None.

See Also v2araplygn, v2ararplygn, v2arwplygn

Map 2D Virtual to Screen V2GPMAP

Format

- LIB Shell: 0x0091 x y
- C Binding: void v2gpmap(x y):
- TMLS Interpreter: v2gpmap x y

Parameters

- FIXPT x: the x-coordinate of the 2D virtual coordinate to be mapped.
- FIXPT y: the y-coordinate of the 2D virtual coordinate to be mapped.

Description

This function maps the 2D virtual coordinates (x, y) to the corresponding physical screen coordinates according to the current window/viewport mapping ratio.

Return Value Screen coordinates are returned in the Read FIFO The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the C Binding's iogetw function to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also v2gpmapx, v2gpmapy, v2gpunmp

V2GPMAPX Map 2D X to Screen

Format

- LIB Shell: 0x0092 x
- C Binding: void v2gpmapx(x);
- TMLS Interpreter: v2gpmapx x

Parameters

- FIXPT x: the 2D virtual x coordinate to be transformed.

Description

This function maps the 2D virtual x-coordinate x to the corresponding screen x-coordinate, according to the current window/viewport mapping ratio.

Return Value The mapped x-coordinate is returned in the Read FIFO. The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

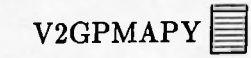
C Binding: Use the C Binding's iogetw function to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also v2gpmap, v2gpmapy, v2gpunmp

Map 2D Y to Screen — V2GPMAPY



Format

- LIB Shell: 0x0093 y
- C Binding: void v2gpmapy(y);
- TMLS Interpreter: v2gpmapy y

Parameters

- FIXPT y: the 2D virtual y-coordinate y to be transformed.

Description

This function maps the 2D virtual y-coordinate y to the corresponding screen y-coordinate, according to the current window/viewport mapping ratio.

Return Value The mapped y-coordinate is returned in the Read FIFO. The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the C Binding's iogetw function to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also v2gpmap, v2gpmapx, v2gpunmp

V2GPSVWP — Set Viewport

Format

- LIB Shell: 0x0090 x_{min} x_{max} y_{min} y_{max}
- C Binding: void v2gpsvwp(x_{min}, x_{max}, y_{min}, y_{max});
- TMLS Interpreter: v2gpsvwp x_{min} x_{max} y_{min} y_{max}

Parameters

- short x_{min}: (Default = 0) the x-coordinate of the left side of the viewport.
- short x_{max}: (Default = 1279, 1023, or 639) the x-coordinate of the right side of the viewport.
- short y_{min}: (Default = 0) the y-coordinate of the top of the viewport.
- short y_{max}: (Default = 1023, 767, or 479) the y-coordinate of the bottom of the viewport.

Description

This function defines the screen viewport area. When you execute this command you also reset the clipping rectangle to the values specified for the viewport.

Note

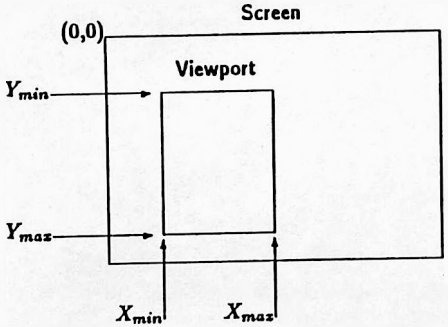
Although this function is part of the 2D layer it uses screen coordinates. The origin is the top left corner of the screen; the y coordinate increments downward.

An OVERFLOW_ERROR will be reported in the Error FIFO if the viewport is set so that:

$$(2)\left(\frac{viewport\_extent}{window\_extent}\right) > (32K - 1)$$

Set Viewport V2GPSVWP

In this case, the vieport or window should be set to another value that does not cause an overflow.



Return Value None.

Error OVERFLOW_ERROR.

See Also v2gpswdw

V2GPSWDW Set Current Window

Format

- LIB Shell: 0x008F x_{min} x_{max} y_{min} y_{max}
- C Binding: void v2gpswdw(x_{min}, x_{max}, y_{min}, y_{max});
- TMLS Interpreter: v2gpswdw x_{min} x_{max} y_{min} y_{max}

Parameters

- FIXPT x_{min}: (Default = -640.0, -512.0, or -320.0) the x-coordinate of the left side of the window.
- FIXPT x_{max}: (Default = 639.0, 511.0, or 319.0) the x-coordinate of the right side of the window.
- FIXPT y_{min}: (Default = -512.0, -384.0, or -240.0) the y-coordinate of the bottom of the window.
- FIXPT y_{max}: (Default = 511.0, 383.0, or 239.0) the y-coordinate of the top of the window.

Description

This function defines the current window. The window is the section of the virtual workspace that is mapped to the screen viewport area. It is set by the most recent call of the v2gpsvwp function. The maximum span in either the horizontal or the vertical direction is 32K. For example, it could go from -16K to 16K in the x-direction and from 0 to 32K in the y-direction. If an overflow occurs, the current window will not change.

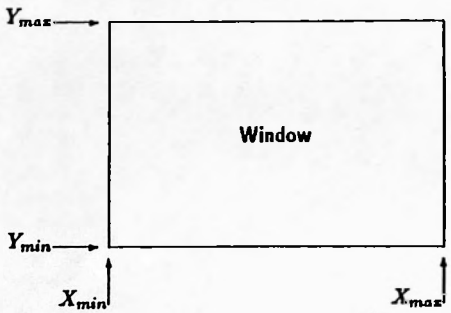
Set Current Window V2GPSWDW

Note

An OVERFLOW_ERROR will be reported in the Error FIFO if the viewport is set so that:

$$(2)(\frac{viewport\_extent}{window\_extent} > (32K - 1)$$

In this case, the viewport or window should be set to another value that does not cause an overflow.



Return Value None.

Error OVERFLOW_ERROR

See Also v2gpsvwp

V2GPUNMP Map Screen Coord. to 2D

Format

- LIB Shell: 0x0094 x y
- C Binding: void v2gpunmp(x, y);
- TMLS Interpreter: v2gpunmp x y

Parameters

- short x: the x-coordinate of the screen coordinate to be transformed.
- short y: the y-coordinate of the screen coordinate to be transformed.

Description

This function maps the 2D physical coordinate (x, y) to the corresponding 2D virtual coordinate, according to the current window/viewport mapping ratio.

Return Value The (x,y) 2D virtual coordinate is returned in the Read FIFO. The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the C Binding's iogetbuf function to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

Error None.

See Also v2gpmap, v2gpmapx, v2gpmapy

Draw Elliptical Arc V2OLEARC

Format

- LIB Shell: 0x00A7 sth eth a b
- C Binding: void v2olearc(sth, eth, a, b);
- TMLS Interpreter: v2olearc sth eth a b

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- FIXPT a: the horizontal virtual radius.
- FIXPT b: the vertical virtual radius.

Description

This function draws a 2D virtual elliptical arc, based on ellipse centered at the 2D current point with radii a and b. The arc is defined by the start angle sth and and the end angle eth. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olellipse, v2olepie, v2oleseg

V2OLELLIPSE — Draw Ellipse

Format

- LIB Shell: 0x00A6 a b
- C Binding: void v2olellipse(a, b);
- TMLS Interpreter: v2olellipse a b

Parameters

- FIXPT a: the horizontal virtual radius.
- FIXPT b: the vertical virtual radius.

Description

This function fills a 2D virtual ellipse with radii a and b centered at the 2D current point. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olearc, v2olepie, v2oleseg

Draw Elliptical Pie Section V2OLEPIE

Format

- LIB Shell: 0x00A8 sth eth a b
- C Binding: void v2olepie(sth, eth, a, b);
- TMLS Interpreter: v2olepie sth eth a b

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- FIXPT a: the horizontal virtual radius.
- FIXPT b: the vertical virtual radius.

Description

This function draws a 2D virtual elliptical pie section, based on an ellipse centered at the 2D current point with radii a and b. The segment is defined by the start angle sth and and the end angle eth. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olearc, v2olellipse, v2oleseg

V2OLESEG Draw Elliptical Segment

Format

- LIB Shell: 0x00A9 sth eth a b
- C Binding: void v2oleseg(sth, eth, a, b);
- TMLS Interpreter: v2oleseg sth eth a b

Parameters

- ANGLE sth: the start angle, measured in the counter-clockwise direction starting at the positive x-axis.
- ANGLE eth: the end angle, measured in the counter-clockwise direction starting at the positive x-axis.
- FIXPT a: the horizontal virtual radius.
- FIXPT b: the vertical virtual radius.

Description

This function draws a 2D virtual elliptical segment, based on an ellipse centered at the 2D current point with radii a and b. The segment is defined by the start angle sth and and the end angle eth. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olearc, v2olellipse, v2olepie

High Precision Ellip. Arc V2OLHPEARC

Format

- LIB Shell: 0x00AE sth eth a b
- C Binding: void v2olhpearc(sth, eth, a, b);
- TMLS Interpreter: v2olhpearc sth eth a b

Parameters

- FIXPT sth: the start angle
- FIXPT eth: the end angle
- FIXPT a: the horizontal radius
- FIXPT b: the vertical radius

Description

This function draws a high precision 2D virtual elliptical arc, based on ellipse centered at the 2D current point with horizontal radius, a, and vertical radius, b. The arc is the part of the ellipse that lies between the start angle, sth, and the end angle, eth. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olearc

V2OLLNTO — Draw 2D Line

Format

- LIB Shell: 0x00A2 x y
- C Binding: void v2ollnto(x, y);
- TMLS Interpreter: v2ollnto x y

Parameters

- FIXPT x: the virtual x-coordinate of the endpoint.
- FIXPT y: the virtual y-coordinate of the endpoint.

Description

This function draws a pixel-wide line from the 2D current position to the endpoint (x, y). The specified endpoint becomes the new 2D current point.

Return Value None.

Error None.

See Also v2olmvto, v2olpt, v2olrlnto, v2olrmvto, scolsmode

Move 2D Current Position V2OLMVTO

Format

- LIB Shell: 0x009F x y
- C Binding: void v2olmvto(x, y);
- TMLS Interpreter: v2olmvto x y

Parameters

- FIXPT x: (Default = 0.0) the virtual x-coordinate of the new position.
- FIXPT y: (Default = 0.0) the virtual y-coordinate of the new position.

Description

This function moves the 2D current position to the point (x, y).

Return Value None.

Error None.

See Also v2ollnto, v2olrlnto, v2olrmvto

V2OLPLINE — Draw Poly-Line

Format

- LIB Shell: 0x00A4 nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v2olpline(nvert, vertptr);
- TMLS Interpreter: v2olpline nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of poly-line vertices.
- V2COORD $vert_i$: the coordinates of the i^{th} poly-line vertex.
- V2COORD *vertptr: a pointer to the buffer containing the poly-line vertices.

Description

This function joins a series of vertices with line segments. Vertices are connected in the order they are specified. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olplygn, v2olrpline, v2olrplygn, scolsmode

Draw a Polygon — V2OLPLYGN

Format

- LIB Shell: 0x00AC nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v2olplygn(nvert, vertptr);
- TMLS Interpreter: v2olplygn nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of polygon vertices.
- V2COORD $vert_i$: the i^{th} polygon vertex.
- V2COORD *vertptr: a pointer to a buffer containing the vertices.

Description

This function joins a series of 2D vertices with line segments, joining the last vertex to the first to close the polygon. Vertices are connected in the order they are specified. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olpline, v2olrpline, v2olrplygn

V2OLPT Draw a Point at Current Point

Format

- LIB Shell: 0x00A1
- C Binding: void v2olpt();
- TMLS Interpreter: v2olpt

Parameters None.

Description

This function sets the pixel located at the 2D current point to the current foreground color. Current drawing attributes are ignored. The current 2D position is not affected by this command.

Return Value None.

Error None.

See Also v2ollnto, v2olmvto, v2olrlnto, v2olrmvto

Draw Rectangle V2OLRECT

Format

- LIB Shell: 0x00AA x y
- C Binding: void v2olrect(x. y):
- TMLS Interpreter: v2olrect x y

Parameters

- FIXPT x: the virtual x-coordinate of a corner of the rectangle.
- FIXPT y: the virtual y-coordinate of a corner of the rectangle.

Description

This function draws a 2D virtual rectangle with opposing corners at the 2D current point (x_o, y_o) and the specified corner (x, y). This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olrrect

V2OLRLNTO Draw 2D Relative Line

Format

- LIB Shell: 0x00A3 deltx delty
- C Binding: void v2olrlnto(deltx, delty);
- TMLS Interpreter: v2olrlnto deltx delty

Parameters

- FIXPT deltx: the virtual Δx value defining the 2D virtual x-coordinate of the end of the line.
- FIXPT delty: the virtual Δy value defining the virtual y-coordinate of the end of the line.

Description

This function draws a pixel-wide line from the 2D current position (x_o, y_o) to the point (x_o + deltx, y_o + delty). The specified endpoint becomes the new 2D current point.

Return Value None.

Error None.

See Also v2olmvto, v2olrlnto, v2olrmvto, scolsmode

Relative Move 2D Curr. Pos. V2OLRMVTO

Format

- LIB Shell: 0x00A0 deltx delty
- C Binding: void v2olrmvto(deltx, delty);
- TMLS Interpreter: v2olrmvto deltx delty

Parameters

- FIXPT deltx: the virtual Δx value defining the virtual x-coordinate of the new position.
- FIXPT delty: the virtual Δy value defining the virtual y-coordinate of the new position.

Description

This function moves the 2D current position (x_o, y_o) to the point $(x_o + \text{deltx}, y_o + \text{delty})$.

Return Value None.

Error None.

See Also v2olInto, v2olmvto, v2olpt, v2olrInto

V2OLRPLINE Draw Relative Poly-Line

Format

- LIB Shell: 0x00A5 ndelt $delt_1$ . . . $delt_{ndelt}$
- C Binding: void v2olrpline(ndelt, deltptr);
- TMLS Interpreter: v2olrpline ndelt $delt_1$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of (Δx, Δy) values defining the vertices of the poly-line.
- V2COORD $delt_i$: the i^{th} (Δx, Δy) delta value. The delta values are displacements from the 2D current point.
- V2COORD *deltptr: a pointer to a buffer containing the delta values.

Description

This function joins a series of vertices with line segments. Vertices are connected in the order they are specified. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olplygn, v2olpline, v2olrplygn, scolsmode

Draw Relative Polygon V2OLRPLYGN

Format

- LIB Shell: 0x00AD ndelt $delt_1$ . . . $delt_{ndelt}$
- C Binding: void v2olrplygn(ndelt, deltptr);
- TMLS Interpreter: v2olrplygn ndelt $delt_1$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of (Δ x, Δ y) values defining the polygon vertices.
- V2COORD $delt_i$: the i^{th} (Δx, Δy) delta value. The delta values are displacements from the 2D current point.
- V2COORD *deltptr: a pointer to the buffer containing the delta values.

Description

This function joins a series of vertices with line segments, joining the last point to the first to close the polygon. Vertices are connected in the order they are specified. This function does not affect the 2D current point.

Return Value None.

Error None.

See Also v2olpline, v2olplygn, v2olrpline

V2OLRRECT Draw Relative Rectangle

Format

- LIB Shell: 0x00AB deltx delty
- C Binding: void v2olrrect(deltx, delty);
- TMLS Interpreter: v2olrrect deltx delty

Parameters

- FIXPT deltx: the Δx value defining a corner of the rectangle.
- FIXPT delty: the Δy value defining a corner of the rectangle.

Description

This function draws a 2D virtual rectangle with opposing corners at the current point (x_o, y_o) and the specified corner $(x_o + \text{deltx}, y_o + \text{delty})$. The 2D current point is not affected.

Return Value None.

Error None.

See Also v2olrect

Set 2D Outline Width V2OLSW

Format

- LIB Shell: 0x0095 new_width
- C Binding: void v2olsw(new_width);
- TMLS Interpreter: scolsw new_width

Parameters

- FIXPT new_width: (Default = -1) the 2D outline width. A zero width indicates a null pen meaning that no outlines will be drawn. A negative width indicates a pixel wide outline is drawn regardless of the current window and viewport. A positive width will be scaled according to the window/viewport ratio when a 2D outline is drawn.

Description

This function sets the new 2D line width, and must be preceded by a scolseshp (Set Endpoint Shape) command.

Return Value None.

Error None.

See Also scolseshp, krhwsils, lsswlapat

V2TXSTRING — Draw 2D Text

Format

- LIB Shell: 0x008E x y length string
- C Binding: void v2txstring(x, y, strptr);
- TMLS Interpreter: v2txstring x y "string"

PARAMETERS:

- FIXPT x: the x-coordinate of the destination point on the screen.
- FIXPT y: the y-coordinate of the destination point on the screen.
- short string[]: the character string. For the TMLS interpreter the string is simply enclosed in single or double quotes. For the LIB Shell send the hexadecimal character codes of the characters in the string.
- unsigned short length: the length of the string in characters.
- short *strptr: a pointer to the character string.

Description

This function draws a string of characters on the screen. It expects strings of word-valued characters. The shell transforms the 2D destination coordinate into a screen coordinate, according to the current window/viewport ratio, and draws the string. The shell does not apply this transformation to the string.

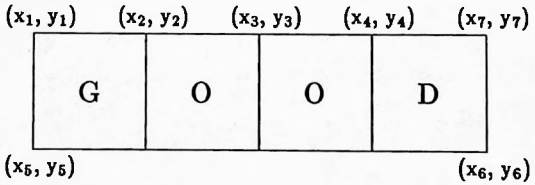
If string drawing mode bit 0 is one (see sctxsdmode), text will not be drawn.

Draw 2D Text — V2TXSTRING

Return Value Values returned in the Read FIFO depend on the string drawing mode. If drawing mode bits 1 and 2 are both 0, there is no return value.

The following table summarizes what is returned for each combination of drawing mode bits 1 and 2. The coordinates refer to the drawing below.

Drawing Mode Bit 2	Drawing Mode Bit 1	Order of coordinates in the Read FIFO.
0	0	None
0	1	(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)
1	0	(x_1, y_1), (x_5, y_5), (x_6, y_6), (x_7, y_7)
1	1	(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4) (x_1, y_1), (x_5, y_5), (x_6, y_6), (x_7, y_7)



The host reads the data as follows:

LIB Shell: Read data from the Read FIFO entrance.

C Binding: Use the C binding iogetbuf function to retrieve the data.

TMLS Interpreter: Use the lsrdfifo utility.

V2TXSTRING — Draw 2D Text

Error None.

See Also sctxsdmode, sctxbextent, sctxbstring, sctxstring, sctxextent

Alternate-Fill Polygon V3ARAPLYGN

Format

- LIB Shell: 0x0113 nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v3araplygn(nvert, vertptr);
- TMLS Interpreter: v3araplygn nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of polygon vertices.
- V3COORD $vert_i$: the i^{th} polygon vertex (x_i, y_i, z_i).
- V3COORD *vert: a pointer to a buffer containing the polygon vertices.

Description

This function draws a filled polygon using the alternate-fill algorithm. It closes the polygon on itself before starting the fill operation.

Return Value None.

Error OVERFLOW_ERROR

See Also v3ararplygn, v3arwplygn, v3arwrplygn

V3ARARPLYGN Alternate-Fill R. Polygon

Format

- LIB Shell: 0x0114 ndelt $delt_1$ . . . $delt_{ndelt}$
- C Binding: void v3ararplygn(ndelt, deltptr);
- TMLS Interpreter: v3ararplygn ndelt $delt_1$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of (Δx, Δy ,Δz) values defining the vertices of the polygon.
- V3COORD $delt_i$: the i^{th} (Δx, Δy ,Δz) delta value. The delta values are displacements from the 3D current point.
- V3COORD *deltptr: a pointer to a buffer containing the delta values.

Description

This function fills the specified polygon using the alternate-fill algorithm. It closes the polygon on itself before starting the fill operation. This function does not affect the 3D current position.

Return Value None

Error OVERFLOW_ERROR

See Also v3araplygn, v3arwplygn, v3arwrplygn

Winding-Number Poly. Fill V3ARWPLYGN

Format

- LIB Shell: 0x0115 nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v3arwplygn(nvert, vertptr);
- TMLS Interpreter: v3arwplygn nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of polygon vertices.
- V3COORD $vert_i$: the i^{th} polygon vertex.
- V3COORD vertptr: a pointer to the buffer containing the polygon vertices.

Description

This function draws a filled polygon using the winding-number fill algorithm. It closes the polygon on itself before starting the fill operation.

Return Value None.

Error OVERFLOW_ERROR

See Also v3araplygn, v3ararplygn, v3arwrplygn

V3ARWRPLYGN Winding No.R.Poly.Fill

Format

- LIB Shell: 0x0116 ndelt $delt_1$ . . . $delt_{ndelt}$
- C Binding: void v3arwrplygn(ndelt, deltptr);
- TMLS Interpreter: v3arwrplygn ndelt $delt_1$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of delta values defining the vertices of the polygon.
- V3COORD $delt_i$: the i^{th} (Δ x, Δy, Δz) value. The delta values are displacements from the 3D current point.
- V3COORD *deltptr: a pointer to the buffer containing the delta values.

Description

This function fills the specified polygon using the winding-number fill algorithm. It closes the specified polygon on itself before starting the fill operation. This function does not affect the 3D current position.

Return Value None.

Error OVERFLOW_ERROR

See Also v3ararplygn, v3araplygn, v3arwplygn

Set Viewing Distance V3GPSDIST

Format

- LIB Shell: 0x00D0 dist
- C Binding: void v3gpsdist(dist);
- TMLS Interpreter: v3gpsdist dist

Parameters

- FIXPT dist: (Default = 500.0) the viewing distance from the eye to the viewing reference point.

Description

This function sets the distance from the eye to the viewing reference point. This attribute only affects 3D drawing.

Return Value None.

Error None.

See Also v3gpsproj, v3vwrpt

V3GPSDISTH Set Hither Plane Distance

Format

- LIB Shell: 0x00D1 dist
- C Binding: void v3gpsdisth(dist);
- TMLS Interpreter: v3gpsdisth dist

Parameters

- FIXPT dist: (Default = -30 000)the distance from the the hither clipping plane to the viewing reference point.

Description

This function sets the distance from the viewing reference point to the hither clipping plane. The hither plane is parallel to the viewplane. To use hither clipping you must enable it with the v3gpshclip function. Hither clipping only affects 3D drawing.

Return Value None.

Error None.

See Also v3gpsdisty, v3gpshclip, v3vwrpt

Set Yon Plane Distance V3GPSDISTY

Format

- LIB Shell: 0x00D2 dist
- C Binding: void v3gpsdisty(dist);
- TMLS Interpreter: v3gpsdisty dist

Parameters

- FIXPT dist: (Default = 30 000) the distance from the yon clipping plane to the viewing reference point.

Description

This function sets the distance from the viewing reference point to the yon clipping plane. To use yon clipping you must enable it with the v3gpsyclip function. The yon clipping plane is parallel to the viewplane. Yon clipping only affects 3D drawing.

Return Value None.

Error None.

See Also v3gpsdisth, v3gpsyclip, v3vwrpt

V3GPSHCLIP Set Hither Clipping Flag

Format

- LIB Shell: 0x00CD flag
- C Binding: void v3gpshclip(flag);
- TMLS Interpreter: v3gpshclip flag

Parameters

- BOOL flag: (Default = null) the enable/disable clipping flag.
  - null: disable hither clipping.
  - non-null: enable hither clipping.

Description

This function enables or disables hither plane clipping. When the hither clipping flag is enabled, no points closer to the viewer than the hither plane are displayed. The hither plane is parallel to the viewplane. Hither clipping only affects 3D drawing.

Return Value None.

Error None.

See Also v3gpsdisth, v3gpsyclip

Set Projection V3GPSPROJ

Format

- LIB Shell: 0x00CF theta
- C Binding: void v3gpsproj(theta);
- TMLS Interpreter: v3gpsproj theta

Parameters

- ANGLE theta: (Default = 60°) the viewing angle. When theta is 0 an orthogonal projection is produced. Otherwise, a perspective projection is produced. The valid range for theta is: $0 \leq$ theta ≤ 179. If theta is outside this range the current projection will not be changed.

Description

This function sets the viewing angle used in 3D to 2D transformations.

Return Value None.

Error OVERFLOW_ERROR

See Also v3gpsdist, v3vwrpt

V3GPSYCLIP Set Yon Clipping Flag

Format

- LIB Shell: 0x00CE flag
- C Binding: void v3gpsyclip(flag);
- TMLS Interpreter: v3gpsyclip flag

Parameters

- BOOL flag: (Default = null) the yon plane clipping flag:
  - null: disable yon clipping.
  - non-null: enable yon clipping.

Description

This function enables or disables yon plane clipping. When the yon clipping flag is enabled, no points farther than the yon clipping plane are displayed. The yon plane is parallel to the viewplane. Yon clipping only affects 3D drawing.

Return Value None.

Error None.

See Also v3gpsdisty, v3gpshclip

Set Modelling Matrix to Identity V3MDID

Format

- LIB Shell: 0x00DE
- C Binding: void v3mdid();
- TMLS Interpreter: v3mdid

Parameters None.

Description:

This function sets the modelling transformation matrix to the identity matrix. Since transformations on the modelling matrix are cumulative, this function may be used to reset the modelling matrix to isolate the effects of particular transformations on an image.

Return Value None.

Error None.

See Also v3mdset

V3MDORG Set Modelling Origin

Format

- LIB Shell: 0x00E0 x y z
- C Binding: void v3mdorg(x, y, z);
- TMLS Interpreter: v3mdorg x y z

Parameters

- FIXPT x: (Default = 0.0) the virtual x-coordinate of the modelling origin.
- FIXPT y: (Default = 0.0) the virtual y-coordinate of the modelling origin.
- FIXPT z: (Default = 0.0) the virtual z-coordinate of the modelling origin.

Description

This function sets the origin section of the modelling transformation to the point (x, y, z). The origin is the centre of rotation, expansion, and contraction for graphic objects.

Return Value None.

Error None.

See Also v3vwrpt

Modelling Rotate X-Axis V3MDROTX

Format

- LIB Shell: 0x00E3 theta
- C Binding: void v3mdrotx(theta);
- TMLS Interpreter: v3mdrotx theta

Parameters

- ANGLE theta: the rotation angle, measured in the counter-clockwise direction in the yz plane.

Description

This function rotates the object about the x-axis by theta degrees. Note that the effect of this function is cumulative on the current modelling matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3mdroty, v3mdrotz, v3mdrotx

V3MDROTY Modelling Rotate Y-Axis

Format

- LIB Shell: 0x00E4 theta
- C Binding: void v3mdroty(theta);
- TMLS Interpreter: v3mdroty theta

Parameters

- ANGLE theta: the rotation angle, measured in the counter-clockwise direction in the xz plane.

Description

This function rotates the object about the y-axis by theta degrees. Note that the effect of this function is cumulative on the current modelling matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3mdrotx, v3mdrotz, v3mdsc, v3mdtran

Modelling Rotate Z-Axis V3MDROTZ

Format

- LIB Shell: 0x00E5 theta
- C Binding: void v3mdrotz(theta);
- TMLS Interpreter: v3mdrotz theta

Parameters

- ANGLE theta: the rotation angle, measured in the counter-clockwise direction in the xy plane.

Description

This function rotates the object about the z-axis by theta degrees. Note that the effect of this function is cumulative on the current modelling matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3mdrotx, v3mdroty, v3mdsc, v3mdtran

V3MDSC Modelling Scale

Format

- LIB Shell: 0x00E1 sx sy sz
- C Binding: void v3mdsc(sx, sy, sz);
- TMLS Interpreter: v3mdsc sx sy sz

Parameters

- FIXPT sx: the 3D scaling component with respect to the x-axis.
- FIXPT sy: the 3D scaling component with respect to the y-axis.
- FIXPT sz: the 3D scaling component with respect to the z-axis.

Description

This function changes the scaling component of the modelling matrix for 3D drawing. Note that the effect of this function is cumulative on the current modelling matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3mdtran, v3mdrotx, v3mdroty, v3mdrotz

Set Current Modelling Matrix V3MDSET

Format

- LIB Shell: 0x00DF $m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,4}$
  $m_{2,1}$ $m_{2,2}$ $m_{2,3}$ $m_{2,4}$
  $m_{3,1}$ $m_{3,2}$ $M_{3,3}$ $m_{3,4}$
  $m_{4,1}$ $m_{4,2}$ $m_{4,3}$ $m_{4,4}$
- C Binding: void v3mdset(m)
- TMLS Interpreter: v3mdset $m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,4}$
  $m_{2,1}$ $m_{2,2}$ $m_{2,3}$ $m_{2,4}$
  $m_{3,1}$ $m_{3,2}$ $m_{3,3}$ $m_{3,4}$
  $m_{4,1}$ $m_{4,2}$ $m_{4,3}$ $m_{4,4}$

Parameters

- FIXPT m[4][4]: (Default = identity matrix) a 4×4 array containing the new modelling matrix data.
- FIXPT $m_{i,j}$: an element in the array of modelling matrix data.

Description

This function replaces the current modelling matrix with the data contained in the 4×4 array m.

Return Value None.

Error OVERFLOW_ERROR

See Also v3mdid, v3vwset, lsinq

V3MDTRAN Modelling Translation

Format

- LIB Shell: 0x00E2 tx ty tz
- C Binding: void v3mdtran(tx, ty, tz);
- TMLS Interpreter: v3mdtran tx ty tz

Parameters

- FIXPT tx: the 3D translation x-component.
- FIXPT ty: the 3D translation y-component.
- FIXPT tz: the 3D translation z-component.

Description

This function adds the vector (tx, ty, tz) to the translation component of the modelling matrix for 3D drawing. Note that the effect of this function is cumulative on the current modelling matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3mdsc, v3mdrotx, v3mdroty, v3mdrotz

Draw 3D Line — V3OLLNTO

Format

- LIB Shell: 0x0103 x y z
- C Binding: void v3ollnto(x, y, z);
- TMLS Interpreter: v3ollnto x y z

Parameters

- FIXPT x: the virtual x-component of the endpoint.
- FIXPT y: the virtual y-component of the endpoint.
- FIXPT z: the virtual z-component of the endpoint.

Description

This function draws a pixel-wide line from the 3D current position to the endpoint (x, y , z). The specified endpoint becomes the new 3D current point.

Return Value None.

Error OVERFLOW_ERROR

See Also v3olmvto, v3olrlnto, v3olrmvto, scolsmode

V3OLMVTO Move 3D Current Position

Format

- LIB Shell: 0x0100 x y z
- C Binding: void v3olmvto(x, y , z);
- TMLS Interpreter: v3olmvto x y z

Parameters

- FIXPT x: (Default = 0.0) the virtual x-coordinate of the new position.
- FIXPT y: (Default = 0.0) the virtual y-coordinate of the new position.
- FIXPT z: (Default = 0.0) the virtual z-coordinate of the new position.

Description

This function moves the 3D current point to the point (x, y, z).

Return Value None.

Error OVERFLOW_ERROR

See Also v3ollnto, v3olrlnto, v3olrmvto

Draw Poly-Line V3OLPLINE

Format

- LIB Shell: 0x0105 nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v3olpline(nvert, vertptr);
- TMLS Interpreter: v3olpline nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of poly-line vertices.
- V3COORD $vert_i$: the coordinates of the i^{th} poly-line vertex.
- V3COORD *vertsptr: a pointer to the buffer containing the poly-line vertices.

Description

This function joins a series of vertices with line segments. Vertices are connected in the order they are specified. This function does not affect the 3D current point.

Return Value None.

Error OVERFLOW_ERROR

See Also v3olplygn, v3olrpline, v3olrplygn, v3olpline

V3OLPLYGN Draw a Polygon

Format

- LIB Shell: 0x0107 nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$
- C Binding: void v3olplygn(nvert, vertptr);
- TMLS Interpreter: v3olplygn nvert $vert_1$ $vert_2$ . . . $vert_{nvert}$

Parameters

- unsigned short nvert: the number of polygon vertices.
- V3COORD $vert_i$: the i^{th} polygon vertex.
- V3COORD *vertptr: a pointer to a buffer containing the vertices.

Description

This function joins a series of 3D vertices with line segments, joining the last vertex to the first to close the polygon. Vertices are connected in the order they are specified. This function does not affect the 3D current point.

Return Value None.

Error None.

See Also v3olpline, v3olrpline, v3olrplygn

Draw 3D Current Point — V3OLPT

Format

- LIB Shell: 0x0102
- C Binding: void v3olpt();
- TMLS Interpreter: v3olpt

Parameters None.

Description

This function sets the pixel located at the 3D current point to the current foreground color. Current drawing attributes are ignored. The current 3D position is not affected by this command.

Return Value None.

Error None.

See Also v3ollnto, v3olmvto, v3olrlnto, v3olrmvto

V3OLRLNTO Draw 3D Relative Line

Format

- LIB Shell: 0x0104 deltx delty deltz
- C Binding: void v3olrlnto(deltx, delty, deltz);
- TMLS Interpreter: v3olrlnto deltx delty deltz

Parameters

- FIXPT deltx: the virtual Δx value defining the 3D virtual x-coordinate of the end of the line.
- FIXPT delty: the virtual Δy value defining the 3D virtual y-coordinate of the end of the line.
- FIXPT deltz: the virtual Δz value defining the 3D virtual z-coordinate of the end of the line.

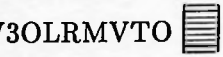
Description

This function draws a pixel-wide line from the 3D current position (x_o, y_o, z_o) to the point $(x_o + \text{deltx}, y_o + \text{delty}, z_o + \text{deltz})$. The specified endpoint becomes the new 3D current point.

Return Value None.

Error None.

See Also v3olmvto, v3olrlnto, v3olrmvto, v3olrlnto

Rel. Move 3D Curr. Pos. V3OLRMVTO 

Format

- LIB Shell: 0x0101 deltx delty deltz
- C Binding: void v3olrmvto(deltx, delty, deltz);
- TMLS Interpreter: v3olrmvto deltx delty deltz

Parameters

- FIXPT deltx: the virtual Δx value defining the virtual x-coordinate of the new position.
- FIXPT delty: the virtual Δy value defining the virtual y-coordinate of the new position.
- FIXPT deltz: the virtual Δz value defining the virtual z-coordinate of the new position.

Description

This function moves the 3D current position from the 3D current position (x_o, y_o, z_o) to the point (x_o + deltx, y_o + delty, z_o + deltz).

Return Value None.

Error None.

See Also v3olpt, v3ollnto, v3olrlnto, v3olmvto

V3OLRPLINE Draw Relative Poly-Line

Format

- LIB Shell: 0x0106 ndelt $delt_1$ $delt_2$ . . . $delt_{ndelt}$
- C Binding: void v3olrpline(ndelt, deltptr);
- TMLS Interpreter: v3olrpline ndelt $delt_1$ $delt_2$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of $(\Delta x_i, \Delta y_i, \Delta z_i)$ values defining the vertices of the poly-line.
- V3COORD $delt_i$: the i^{th} $(\Delta x_i, \Delta y_i, \Delta z_i)$ value. The delta values are displacements from the 3D current point.
- V3COORD *deltptr: a pointer to a buffer containing the delta values.

Description

This function joins a series of vertices with line segments. Vertices are connected in the order they are specified. This function does not affect the 3D current point.

Return Value None.

Error None.

See Also v3olplygn, v3olpline, v3olrplygn, scolsmode

Draw a Relative Polygon V3OLRPLYGN

Format

- LIB Shell: 0x0108 ndelt $delt_1$ . . . $delt_{ndelt}$
- C Binding: void v3olrplygn(ndelt, deltptr);
- TMLS Interpreter: v3olrplygn ndelt $delt_1$ . . . $delt_{ndelt}$

Parameters

- unsigned short ndelt: the number of (Δx_i, Δy_i, Δz_i) values defining the polygon vertices.
- V3COORD $delt_i$: the i^{th} delta value (Δx_i, Δy_i, Δz_i). The delta values are displacements from the 3D current point.
- V3COORD *deltptr: a pointer to a buffer containing the delta values.

Description

This function joins a series of 3D vertices with line segments, joining the last vertex to the first to close the polygon. Vertices are connected in the order they are specified. This function does not affect the 3D current point.

Return Value None.

Error None.

See Also v3olpline, v3olplygn, v3olrpline

V3VWID Set Viewing Matrix to Identity

Format

- LIB Shell: 0x00F0
- C Binding: void v3vwid();
- TMLS Interpreter: v3vwid

Parameters None.

Description

This function sets the viewing transformation matrix to the identity matrix. Since transformations on the viewing matrix are cumulative, this function may be used to reset the viewing matrix to isolate the effects of particular transformations on an image.

Return Value None.

Error None.

See Also v3vwset

Viewing Rotate X-Axis V3VWROTX

Format

- LIB Shell: 0x00F3 theta
- C Binding: void v3vwrotx(theta);
- TMLS Interpreter: v3vwrotx theta

Parameters

- ANGLE theta: the rotation angle, measured in the counter-clockwise direction in the yz plane.

Description

This function rotates the position of the viewer on the viewing sphere by theta degrees with respect to the x-axis. The center of the viewing sphere is the viewer reference point specified with the v3vwrpt function. Note that the effect of this function is cumulative on the current viewing matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3vwroty, v3vwrotz

V3VWROTY Viewing Rotate Y-Axis

Format

- LIB Shell: 0x00F4 theta
- C Binding: void v3vwroty(theta);
- TMLS Interpreter: v3vwroty theta

Parameters

- ANGLE theta: the rotation angle, measured in the counter-clockwise direction in the xz plane.

Description

This function rotates the position of the viewer on the viewing sphere by theta degrees with respect to the y-axis. The center of the viewing sphere is the viewer reference point specified with the v3vwrpt function. Note that the effect of this function is cumulative on the current viewing matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3vwrotx, v3vwrotz

Viewing Rotate Z-Axis V3VWROTZ

Format

- LIB Shell: 0x00F5 theta
- C Binding: void v3vwrotz(theta);
- TMLS Interpreter: v3vwrotz theta

Parameters

- ANGLE theta: the rotation angle, measured in the counter-clockwise direction in the xy plane.

Description

This function rotates the position of the viewer on the viewing sphere by theta degrees with respect to the z-axis. The center of the viewing sphere is the viewer reference point specified with the v3vwrpt function. Note that the effect of this function is cumulative on the current viewing matrix.

Return Value None.

Error OVERFLOW_ERROR

See Also v3vwrotx, v3vwroty

V3VWRPT Set Viewing Reference Point

Format

- LIB Shell: 0x00F2 x y z
- C Binding: void v3vwrpt(x, y, z);
- TMLS Interpreter: v3vwrpt x y z

Parameters

- FIXPT x: (Default = 0.0) the x-coordinate of the new viewing reference point.
- FIXPT y: (Default = 0.0) the y-coordinate of the new viewing reference point.
- FIXPT z: (Default = 0.0) the z-coordinate of the new viewing reference point.

Description

This function sets the viewing reference point. The viewing reference point is the point that the user looks at in the centre of the field of view.

Return Value None.

Error None.

See Also v3mdorg

Set Current Viewing Matrix V3VWSET

Format

- LIB Shell: 0x00F1 $v_{1,1}$ $v_{1,2}$ $v_{1,3}$ $v_{1,4}$
  $v_{2,1}$ $v_{2,2}$ $v_{2,3}$ $v_{2,4}$
  $v_{3,1}$ $v_{3,2}$ $v_{3,3}$ $v_{3,4}$
  $v_{4,1}$ $v_{4,2}$ $v_{4,3}$ $v_{4,4}$
- C Binding: void v3vwset(v);
- TMLS Interpreter: v3vwset $v_{1,1}$ $v_{1,2}$ $v_{1,3}$ $v_{1,4}$
  $v_{2,1}$ $v_{2,2}$ $v_{2,3}$ $v_{2,4}$
  $v_{3,1}$ $v_{3,2}$ $v_{3,3}$ $v_{3,4}$
  $v_{4,1}$ $v_{4,2}$ $v_{4,3}$ $v_{4,4}$

Parameters

- FIXPT v[4][4]: (Default = identity matrix) a 4 × 4 array containing the new viewing matrix data.
- FIXPT $v_{i,j}$: an element in the array of viewing matrix data.

Description

This function replaces the current viewing matrix with the data contained in the 4 × 4 array v.

Return Value None.

Error OVERFLOW_ERROR

See Also v3vwid, v3mdset, lsinq

Appendix A

Default Parameter Values

Default system parameters and LUT contents are summarized in this appendix. Parameters are listed with respect to the command which sets them. System dependent parameters are marked with a †. Contents of the default LUT are given in the last figure.

A.1 Default System Parameters

COMMAND:	VALUE:	DESCRIPTION:
krgcmv	0 0	initial cursor position is (0,0)
krgcsda	1	first pass transparency is source-driven
	0xFFFF	first pass plane mask
	0	first pass foreground color is black
	0	first pass background color is black
	0	first pass boolean operator is REPLACE
	1	second pass transparency is source-driven
	0xFFFF	second pass plane mask
	255	second pass foreground color is white
	0	second pass background color is black
	0	second pass boolean operator is REPLACE
	0	clip cursor to current clipping rectangle
krgcsdm	0	set cursor mode to restore
krhwsbcol	0	background color set to index is black
krhwsbop	0	boolean operator set to replace
krhwsclipr†	0 0 1279 1023	clip rectangle is (0,0) to (1279,1023)
	0 0 1023 767	clip rectangle is (0,0) to (1023,767)
	0 0 639 479	clip rectangle is (0,0) to (639,479)
krhwsfcol	255	foreground color set to index is white
krhwsils	0xFFFF	line style is solid
krhwsmsk	0xFFFF	plane mask = all planes enabled

COMMAND:	VALUE:	DESCRIPTION:
krhwssmsk	0xFFFF	seek mask = all planes enabled
krhwsscol	0	sought color index is black
krhwstdef	0	transparency mode set to opaque
kriossmode	0	no FIFO data swapping
lsclrealloc	1	memory reallocation enabled
lsclsblkss	20480	20 Kwords
lserror	0	error reporting is off
lslutsl	0 0	logical LUT 0 mapped to physical LUT 0
lspkdi	–	disable pick mode
lspksidl	0	pick ID low word
lspksidh	0	pick ID high word
lssbm	-1	the frame buffer
lssfont	-1	font is default raster font
lssgc	-1	an arrow cursor
lssafapat	-1	area fill pattern is solid
lsswlapat	-1	wide line fill pattern is solid
scarspin	0	disable pattern pinning to screen origin
scolmvto	0 0	current point is origin
scolseshp	0 0	squared off start and end shapes
scolsmode	0x0100	draw endpoints; no line style continuity
scolsw	1	line width is 1
scsorg	0 0	screen origin is top left corner
scsthincr	5	theta increment is 5°
sctxsalign	0 0	alignment is top left justified
sctxsbf	0	boldening disabled
sctxsblrot	0	baseline rotation is 0°
sctxsbx	0.0	no extra space to spread between words

DEFAULT PARAMETER VALUES

COMMAND:	VALUE:	DESCRIPTION:
sctxschrot	0	cell rotation is 0°
sctxsim	0	italic disabled
sctxsincs	0	inter-character spacing is 0
sctxslm	0	lightening disabled
sctxssf	1.0 1.0	scaling factor is 1 in x and y directions
sctxssow	0	strike out disabled
sctxssw	1	stroke width = 1
sctxsuw	0	underlining disabled
sctxswpw	0	disable word placement width
v2gpsvwp†	0 1279 0 1023	viewport is visible screen
	0 0 1023 767	viewport is visible screen
	0 0 639 479	viewport is visible screen
v2gpswdw†	-640.0 639.0	window is (-640.0,-512.0) to (639.0,
	-512.0 511.0	511.0) or
	-512.0 511.0	window is (-512.0,-384.0) to (511.0,
	-384.0 383.0	383.0) or
	-320.0 319.0	window is (-320.0,-240.0) to (319.0,
	-512.0 511.0	239.0)
v2olmvto	0.0 0.0	2D current point is (0,0)
v2olsw	-1	pixel-wide 2D lines
v3gpsdist	500.0	viewing distance is 500
v3gpsdisth	-30000.0	hither distance
v3gpsdisty	30000.0	yon distance
v3gpshclip	0	hither clipping is off
v2gpsproj	60	projection angle is 60°

COMMAND:	VALUE:	DESCRIPTION:
v3gpsyclip	0	yon clipping is off
v3mdorg	0.0 0.0 0.0	modelling origin is (0,0,0)
v3mdset	–	set modelling matrix to identity
v3olmvto	0.0 0.0 0.0	3D current point is (0,0,0)
v3vwrpt	0.0 0.0 0.0	viewing reference point is (0,0,0)
v3vwset	–	set viewing matrix to identity

A.2 Default LUT Contents

This section contains the contents of the default Lookup Table (label = -1). The following figure contains three decimal numbers per entry, in the format of the *LUTVAL* data structure.

DEFAULT PARAMETER VALUES

	red:	green:	blue:
Entry 0 :	0,	0,	0
Entry 1 :	16,	16,	16
Entry 2 :	32,	32,	32
Entry 3 :	48,	48,	48
Entry 4 :	64,	64,	64
Entry 5 :	80,	80,	80
Entry 6 :	96,	96,	96
Entry 7 :	112,	112,	112
Entry 8 :	128,	128,	128
Entry 9 :	144,	144,	144
Entry 10 :	160,	160,	160
Entry 11 :	176,	176,	176
Entry 12 :	192,	192,	192
Entry 13 :	208,	208,	208
Entry 14 :	224,	224,	224
Entry 15 :	240,	240,	240
Entry 16 :	0,	0,	0
Entry 17 :	32,	0,	0
Entry 18 :	64,	0,	0
Entry 19 :	96,	0,	0
Entry 20 :	128,	0,	0
Entry 21 :	160,	0,	0
Entry 22 :	192,	0,	0
Entry 23 :	224,	0,	0
Entry 24 :	240,	0,	0
Entry 25 :	240,	32,	32
Entry 26 :	240,	64,	64
Entry 27 :	240,	96,	96
Entry 28 :	240,	128,	128
Entry 29 :	240,	160,	160
Entry 30 :	240,	192,	192
Entry 31 :	240,	224,	224
Entry 32 :	0,	0,	0
Entry 33 :	32,	0,	16
Entry 34 :	64,	0,	32
Entry 35 :	96,	0,	48
Entry 36 :	128,	0,	64
Entry 37 :	160,	0,	80
Entry 38 :	192,	0,	96
Entry 39 :	224,	0,	112
Entry 40 :	240,	0,	128
Entry 41 :	240,	32,	144
Entry 42 :	240,	64,	160
Entry 43 :	240,	96,	176
Entry 44 :	240,	128,	192
Entry 45 :	240,	160,	208
Entry 46 :	240,	192,	224
Entry 47 :	240,	224,	240
Entry 48 :	0,	0,	0
Entry 49 :	32,	0,	32
Entry 50 :	64,	0,	64
Entry 51 :	96,	0,	96
Entry 52 :	128,	0,	128
Entry 53 :	160,	0,	160
Entry 54 :	192,	0,	192
Entry 55 :	224,	0,	224
Entry 56 :	240,	0,	240
Entry 57 :	240,	32,	240
Entry 58 :	240,	64,	240
Entry 59 :	240,	96,	240
Entry 60 :	240,	128,	240
Entry 61 :	240,	160,	240
Entry 62 :	240,	192,	240
Entry 63 :	240,	224,	240
Entry 64 :	0,	0,	0
Entry 65 :	16,	0,	32
Entry 66 :	32,	0,	64
Entry 67 :	48,	0,	96
Entry 68 :	64,	0,	128
Entry 69 :	80,	0,	160
Entry 70 :	96,	0,	192
Entry 71 :	112,	0,	224
Entry 72 :	128,	0,	240
Entry 73 :	144,	32,	240
Entry 74 :	160,	64,	240
Entry 75 :	176,	96,	240

Entry 76 :	192,	128,	240	Entry 115 :	0,	96,	96
Entry 77 :	208,	160,	240	Entry 116 :	0,	128,	128
Entry 78 :	224,	192,	240	Entry 117 :	0,	160,	160
Entry 79 :	240,	224,	240	Entry 118 :	0,	192,	192
Entry 80 :	0,	0,	0	Entry 119 :	0,	224,	224
Entry 81 :	0,	0,	32	Entry 120 :	0,	240,	240
Entry 82 :	0,	0,	64	Entry 121 :	32,	240,	240
Entry 83 :	0,	0,	96	Entry 122 :	64,	240,	240
Entry 84 :	0,	0,	128	Entry 123 :	96,	240,	240
Entry 85 :	0,	0,	160	Entry 124 :	128,	240,	240
Entry 86 :	0,	0,	192	Entry 125 :	160,	240,	240
Entry 87 :	0,	0,	224	Entry 126 :	192,	240,	240
Entry 88 :	0,	0,	240	Entry 127 :	224,	240,	240
Entry 89 :	32,	32,	240				
Entry 90 :	64,	64,	240				
Entry 91 :	96,	96,	240	Entry 128 :	0,	0,	0
Entry 92 :	128,	128,	240	Entry 129 :	0,	32,	16
Entry 93 :	160,	160,	240	Entry 130 :	0,	64,	32
Entry 94 :	192,	192,	240	Entry 131 :	0,	96,	48
Entry 95 :	224,	224,	240	Entry 132 :	0,	128,	64
Entry 96 :	0,	0,	0	Entry 133 :	0,	160,	80
Entry 97 :	0,	16,	32	Entry 134 :	0,	192,	96
Entry 98 :	0,	32,	64	Entry 135 :	0,	224,	112
Entry 99 :	0,	48,	96	Entry 136 :	0,	240,	128
Entry 100 :	0,	64,	128	Entry 137 :	32,	240,	144
Entry 101 :	0,	80,	160	Entry 138 :	64,	240,	160
Entry 102 :	0,	96,	192	Entry 139 :	96,	240,	176
Entry 103 :	0,	112,	224	Entry 140 :	128,	240,	192
Entry 104 :	0,	128,	240	Entry 141 :	160,	240,	208
Entry 105 :	32,	144,	240	Entry 142 :	192,	240,	224
Entry 106 :	64,	160,	240	Entry 143 :	224,	240,	240
Entry 107 :	96,	176,	240	Entry 144 :	0,	0,	0
Entry 108 :	128,	192,	240	Entry 145 :	0,	32,	0
Entry 109 :	160,	208,	240	Entry 146 :	0,	64,	0
Entry 110 :	192,	224,	240	Entry 147 :	0,	96,	0
Entry 111 :	224,	240,	240	Entry 148 :	0,	128,	0
Entry 112 :	0,	0,	0	Entry 149 :	0,	160,	0
Entry 113 :	0,	32,	32	Entry 150 :	0,	192,	0
Entry 114 :	0,	64,	64	Entry 151 :	0,	224,	0

DEFAULT PARAMETER VALUES

Entry 152 :	0,	240,	0	Entry 191 :	240,	240,	224
Entry 153 :	32,	240,	32	Entry 192 :	0,	0,	0
Entry 154 :	64,	240,	64	Entry 193 :	32,	16,	0
Entry 155 :	96,	240,	96	Entry 194 :	64,	32,	0
Entry 156 :	128,	240,	128	Entry 195 :	96,	48,	0
Entry 157 :	160,	240,	160	Entry 196 :	128,	64,	0
Entry 158 :	192,	240,	192	Entry 197 :	160,	80,	0
Entry 159 :	224,	240,	224	Entry 198 :	192,	96,	0
Entry 160 :	0,	0,	0	Entry 199 :	224,	112,	0
Entry 161 :	16,	32,	0	Entry 200 :	240,	128,	0
Entry 162 :	32,	64,	0	Entry 201 :	240,	144,	32
Entry 163 :	48,	96,	0	Entry 202 :	240,	160,	64
Entry 164 :	64,	128,	0	Entry 203 :	240,	176,	96
Entry 165 :	80,	160,	0	Entry 204 :	240,	192,	128
Entry 166 :	96,	192,	0	Entry 205 :	240,	208,	160
Entry 167 :	112,	224,	0	Entry 206 :	240,	224,	192
Entry 168 :	128,	240,	0	Entry 207 :	240,	240,	224
Entry 169 :	144,	240,	32	Entry 208 :	0,	0,	0
Entry 170 :	160,	240,	64	Entry 209 :	16,	0,	0
Entry 171 :	176,	240,	96	Entry 210 :	48,	16,	16
Entry 172 :	192,	240,	128	Entry 211 :	64,	16,	16
Entry 173 :	208,	240,	160	Entry 212 :	96,	32,	32
Entry 174 :	224,	240,	192	Entry 213 :	112,	32,	32
Entry 175 :	240,	240,	224	Entry 214 :	144,	48,	48
Entry 176 :	0,	0,	0	Entry 215 :	160,	48,	48
Entry 177 :	32,	32,	0	Entry 216 :	192,	64,	64
Entry 178 :	64,	64,	0	Entry 217 :	192,	80,	80
Entry 179 :	96,	96,	0	Entry 218 :	208,	112,	112
Entry 180 :	128,	128,	0	Entry 219 :	208,	128,	128
Entry 181 :	160,	160,	0	Entry 220 :	224,	160,	160
Entry 182 :	192,	192,	0	Entry 221 :	224,	176,	176
Entry 183 :	224,	224,	0	Entry 222 :	240,	208,	208
Entry 184 :	240,	240,	0	Entry 223 :	240,	224,	224
Entry 185 :	240,	240,	32	Entry 224 :	0,	0,	0
Entry 186 :	240,	240,	64	Entry 225 :	0,	16,	0
Entry 187 :	240,	240,	96	Entry 226 :	16,	48,	16
Entry 188 :	240,	240,	128	Entry 227 :	16,	64,	16
Entry 189 :	240,	240,	160	Entry 228 :	32,	96,	32
Entry 190 :	240,	240,	192	Entry 229 :	32,	112,	32

```
Entry 230 : 48,  144,   48
Entry 231 : 48,  160,   48
Entry 232 : 64,  192,   64
Entry 233 : 80,  192,   80
Entry 234 : 112, 208,  112
Entry 235 : 128, 208,  128
Entry 236 : 160, 224,  160
Entry 237 : 176, 224,  176
Entry 238 : 208, 240,  208
Entry 239 : 224, 240,  224
Entry 240 : 0,   0,      0
Entry 241 : 0,   0,     16
Entry 242 : 16,  16,    48
Entry 243 : 16,  16,    64
Entry 244 : 32,  32,    96
Entry 245 : 32,  32,   112
Entry 246 : 48,  48,   144
Entry 247 : 48,  48,   160
Entry 248 : 64,  64,   192
Entry 249 : 80,  80,   192
Entry 250 : 112, 112,  208
Entry 251 : 128, 128,  208
Entry 252 : 160, 160,  224
Entry 253 : 176, 176,  224
Entry 254 : 208, 208,  240
Entry 255 : 224, 224,  240
```

Appendix B

Drawing Attributes

DRAWING ATTRIBUTES

All drawing operations respect the current general drawing attributes:

General Attributes	
Attribute	Command Responsible
transparency definition	krhwstdef
foreground color	krhwsfcol
background color	krhwsbcol
plane mask	krhwsmsk
boolean operator	krhwsbop
clipping rectangle	krhwsclipr

Any exceptions are noted in the appropriate command description.

In addition to the general attributes, outlines drawn with screen layer commands or higher have their own special attributes:

Outline Attributes	
Attribute	Command Responsible
initial line style	krhwsils
current line style	krhwscls
line style continuity	scolsmode
screen line width	scolsw
line start/end shape	scolseshp
wide line area pattern	lsswlapat
theta angle increment	scsthincr
2D line width	v2olsw

The theta angle is used for drawing:

- screen layer outlines having line widths greater than one.
- filled primitives with one or more curved edges.
- large elliptic outlines.
- large elliptic arcs.

All area fills performed in the screen layer, or higher, follow the current area fill pattern. The command to select the current area fill pattern is lssafapat.

Appendix C

Summary of Commands

C.1 Transforms

Screen Transforms

NAME	DESCRIPTION
scsorg	set the screen origin
lsdclpl	define a multi-clipping rectangle list
lsrclpl	remove a multi-clipping rectangle list
lssclpl	select a multi-clipping rectangle list

2D Transforms

NAME	DESCRIPTION
v2gpsvwp	set the current viewport
v2gpswdw	set the current window
v2gpmap	map 2D virtual coordinate to screen coordinate
v2gpmapx	map 2D virtual x-coordinate to screen x-coordinate
v2gpmapy	map 2D virtual y-coordinate to screen y-coordinate
v2gpunmp	map screen coordinate to 2D virtual coordinate

3D Transforms

NAME	DESCRIPTION
v3gpsdist	set the distance from the eye to the viewing reference point
v3gpsdisth	set the distance from the hither plane to viewing reference point
v3gpsdisty	set the distance from the yon plane to viewing reference point
v3gpshclip	enable/disable hither clipping
v3gpsproj	set the viewing angle for the 2D and 3D transformations
v3gpsyclip	enable/disable yon clipping

Modelling Transforms

NAME	DESCRIPTION
v3mdid	reset the modelling transformation matrix to identity
v3mdorg	define the modelling origin
v3mdrotx	rotate the model around the x-axis
v3mdroty	rotate the model around the y-axis
v3mdrotz	rotate the model around the z-axis
v3mdsc	scale the model
v3mdset	load the modelling matrix directly
v3mdtran	translate the model

Viewing Transforms

NAME	DESCRIPTION
v3vwid	set the viewing matrix to identity
v3vwrotx	rotate the view of the model with respect to the x-axis
v3vwroty	rotate the view of the model with respect to the y-axis
v3vwrotz	rotate the view of the model with respect to the z-axis
v3vwrpt	set the viewing reference point
v3vwset	load the viewing matrix directly

C.2 LUTs

NAME	DESCRIPTION
lslutsl	select a LUT
lslutmd	modify a LUT
lslutrd	read a LUT
lslutrs	reset specified LUT to default values
lslutpm	set LUT plane mask
lslutblink	set LUT blinking
lslutsblink	stop all blinking

C.3 The Drawing Pen

NAME	DESCRIPTION
scolmvto	define the start of a line drawing
v2olmvto	move the lineto-moveto position to the 2D current point
v2olrmvto	relative move the lineto-moveto position to the 2D current point
v3olmvto	move cursor to 3D current point
v3olrmvto	move cursor relative to 3D current point

C.4 The Graphics Cursor

NAME	DESCRIPTION
lsdgc	define the graphics cursor shape
krgcfr	freeze the graphics cursor
krgcsdm	set cursor display mode
lsrgc	remove cursor bit map shape
lssgc	select the cursor bit map shape
krgcdi	disable graphics cursor drawing
krgcen	enable graphics cursor drawing
krgcmv	move graphics cursor
krgcsda	set cursor drawing attributes

C.5 Outlines

Outline Attributes

NAME	DESCRIPTION
scolseshp	set outline endpoint shape
scolsw	set screen outline width
scsthincr	set a theta increment
lsswlapat	set wide line fill pattern
krhwsils	set the initial line style
krhwgcls	get current line style
krhwscls	set current line style
krhwrcls	reset current line style to the initial line style
scolsmode	enable/disable line style continuity
v2olsw	set 2D outline width

Direct Screen Outlines

NAME	DESCRIPTION
scollnto	draw a screen line
scolplygn	draw a polygon
scolearc	draw an elliptical arc
scolellipse	draw an ellipse
scolepie	draw an elliptical pie section
scoleseg	draw an elliptical segment
scolhspline	draw a hermite-type spline
scollnto	draw a line
scolpline	draw a poly-line
scolrect	draw a rectangle
scolrrect	draw a rounded rectangle
scolhpearc	draw a high precision elliptic arc

2D Virtual Outlines

NAME	DESCRIPTION
v2olearc	draw a 2D elliptic arc
v2olellipse	draw a 2D ellipse
v2olepie	draw a 2D elliptic pie section
v2oleseg	draw a 2D elliptic segment
v2olhpearc	draw a high precision 2D elliptic arc
v2ollnto	draw a 2D pixel-wide line
v2olpline	draw a 2D poly-line
v2olplygn	draw a 2D polygon
v2olpt	draw a pixel
v2olrect	draw a 2D rectangle
v2olrlnto	draw a relative pixel-wide line
v2olrpline	draw a relative 2D poly-line
v2olrplygn	draw a relative 2D polygon
v2olrrect	draw a relative 2D rectangle

3D Virtual Outlines

NAME	DESCRIPTION
v3ollnto	draw a 3D pixel-wide line
v3olpline	draw a 3D poly-line
v3olplygn	draw a 3D polygon
v3olpt	draw a point (a pixel)
v3olrlnto	draw a relative 3D pixel-wide line
v3olrpline	draw a 3D relative poly-line
v3olrplygn	draw a relative 3D polygon

C.6 Fills

Area Attributes

NAME	DESCRIPTION
lsdapat	define a fill pattern
lssafapat	select the current area fill pattern
lsswlapat	select the current wide line fill pattern
lsrapat	remove a fill pattern
scarspin	enable pattern pinning
lsbmarmsk	select area fill pattern mask

Primitive Clears

NAME	DESCRIPTION
krhwsclr	clear the visible screen to specified color
krhwcclr	clear the current clipping rectangle
krhwrclr	clear a rectangle to a specified color
krhwfclr	clear frame buffer to specified color

Direct Screen Fills

NAME	DESCRIPTION
scaraplygn	fill alternate-fill polygon
scarbfill	arbitrary area boundary fill
scarellipse	fill ellipse
scarepie	fill elliptical pie section
scareseg	fill elliptical segment
scarffill	arbitrary area flood fill
scarrect	fill rectangle
scarrrect	fill rounded rectangle
scarwplygn	fill winding-number polygon
scartri	fill triangle

2D Virtual Fills

NAME	DESCRIPTION
v2araplygn	alternate-fill a 2D polygon
v2ararplygn	alternate-fill a relative 2D polygon
v2arbfill	boundary fill a 2D arbitrary area
v2arellipse	fill a 2D ellipse
v2arepie	fill a 2D elliptic pie section
v2areseg	fill a 2D elliptic segment
v2arffill	flood fill a 2D arbitrary area
v2arrect	fill a 2D rectangle
v2arrrect	fill a rounded 2D rectangle
v2arwplygn	winding-number fill a 2D polygon
v2arwrplygn	winding-number fill a relative 2D polygon

3D Virtual Fills

NAME	DESCRIPTION
v3araplygn	alternate-fill a 3D polygon
v3ararplygn	alternate-fill a relative 3D polygon
v3arwplygn	winding-number fill a 3D polygon
v3arwrplygn	winding-number fill a relative 3D polygon

C.7 Text

Drawing Text

NAME	DESCRIPTION
lsdfont	define a text font
lsrfont	remove a font
lssfont	select the current font
sctxstring	draw text on the screen
sctxsdmode	select text drawing mode
v2txstring	draw 2D text on screen

Text Attributes

NAME	DESCRIPTION
sctxsuw	set underline attributes
sctxssow	set strike-out thickness
sctxsim	set italicization mask
sctxslm	set lightening mask
sctxsbf	set bolding factor
sctxssw	set stroke width

Scaling and Rotating Text

NAME	DESCRIPTION
sctxssf	set character size scaling factors
sctxsblrot	set text string baseline rotation angle
sctxschrot	set character cell rotation angle

Word Placement

NAME	DESCRIPTION
lsswpvar	set word placement variable
sctxextent	determine the extent of a string
sctxsalign	select string alignment operation
sctxsbx	set extra intercharacter spacing
sctxsincs	set intercharacter spacing
sctxswpw	set word placement overall width and disable/enable word placement

C.8 Command Lists

NAME	DESCRIPTION
lsclappend	append command lists
lsclsblksz	set command list memory block size
lsclrealloc	set command list reallocation flag
lsclclose	close a command list definition
lscldel	delete a command list
lsclexec	execute a command list
lsclloop	execute a command list a specified number of times
lsclmodify	modify a command list
lsclopen	open a command list
lsclpos	return word offset from start of command list
lsclread	read a command list
lsclcompr	compress a command list

C.9 Drawing Attributes

NAME	DESCRIPTION
krhwsbop	set the current graphics boolean operator
krhwsclipr	set clipping rectangle boundaries
krhwsfcm	set the flash clear mode
krhwsmsk	set the plane mask
krhwstdef	set transparency operation mode
krhwsdirect	set the window access mode
krhwsbcol	set background expansion color
krhwsfcol	set the foreground expansion color

C.10 Object and Layer Tagging

NAME	DESCRIPTION
lstgobj	start an object definition
lstglay	start a layer definition
lstgrobj	delete an object
lstgrlay	delete a layer
lstgslvis	set layer visibility
lstgsvtbl	set layer visibility table

C.11 Primitive Raster Operations

Raster Read and Writes

NAME	DESCRIPTION
lsrdrst	raster read
lsrlrdrst	run length raster read
lsrlwrrst	run length raster write
lswrrst	raster write
krhwgpix	return a specified pixel value
krhwspix	set the pixel on the screen

Raster Copies

NAME	DESCRIPTION
krhwcprss	copy a bit map with rotation
krhwcpss	copy a bit map
krhwcpxmss	copy a bit map with x-mirror transform
krhwcpymss	copy a bit map with y-mirror transform

Pixel Seek Operations

NAME	DESCRIPTION
krhwsscol	set the pixel-seek color
krhwssmsk	set the pixel-seek plane mask
krhwubps	performs an "until-backward" pixel seek
krhwufps	performs an "until-forward" pixel seek
krhwwbps	performs a "while-backward" pixel seek
krhwwfps	performs a "while-forward" pixel seek

C.12 Picking

NAME	DESCRIPTION
lspken	enable pick mode
lspkdi	disable pick mode
lspksidl	set pick ID low word
lspksidh	set pick ID high word

C.13 Bit Map Storage Areas

NAME	DESCRIPTION
lsrbm	remove bit map storage area
lscbm	create bit map storage area
lssbm	select bit map storage area
lsfbm	fill bit map storage area
lscpbmsm	copy a bit map from the screen to the current bit map storage area
lscpbmms	copy a bit map from the current bit map storage area to the screen
lscpbmzm	copy a bit map with zoom into the frame buffer

C.14 System Inquiries

NAME	DESCRIPTION
lsinq	inquire current shell parameters
lsinqgcp	inquire current cursor position
lsinqsys	inquire current system parameters
lsinqmem	inquire amount of available Local RAM

C.15 Miscellaneous

NAME	DESCRIPTION
kriossmode	Write FIFO bit map data swap mode
lsnoop	no operation
lswait	wait a specified number of frames
lserror	enable/disable error reporting
lsskip	skip words in command list
lsemulen†	enable graphics adapter emulation mode
lsegasw‡	select input to EGA switch
lsvgasw‡	select input to VGA switch
lsvsint	interrupt host on VSYNC

† — PG-Series-1 and PG2-Series-1 only
‡ — PG-Series-1 only

Appendix D

LIB Shell Opcodes

This appendix consists of two sections. The first section lists command opcodes in numerical order. The second section contains all the commands in alphabetical order, along with their respective opcodes.

D.1 Opcodes in Numerical Order

Name:	Hex Opcode:	Decimal Opcode:
LSCLOPEN	0x0001	1
LSCLCLOSE	0x0002	2
LSCLEXEC	0x0003	3
LSCLLOOP	0x0004	4
LSCLDEL	0x0005	5
LSCLREAD	0x0006	6
LSCLAPPEND	0x0007	7
LSCLMODIFY	0x0008	8
LSCLSBLKSZ	0x0009	9
LSCLREALLOC	0x000A	10
LSINQMEM	0x000B	11
LSCLPOS	0x000C	12
KRHWDIAG	0x000F	15
KRHWSFCM	0x0010	16
KRHWSZP	0x0011	17
KRIOSSMODE	0x0012	18
LSERROR	0x0013	19
KRHWGCLS	0x0014	20
KRHWSCLS	0x0015	21
KRHWRCLS	0x0016	22
KRHWSDIRECT	0x0017	23
KRHWSTDEF	0x0018	24
KRHWSFCOL	0x0019	25
KRHWSBCOL	0x001A	26
KRHWSBOP	0x001B	27
KRHWSMSK	0x001C	28
KRHWSSMSK	0x001D	29
KRHWSCLIPR	0x001E	30
LSWRRST	0x001F	31
LSRDRST	0x0020	32
KRHWFCLR	0x0021	33
KRHWSCLR	0x0022	34
KRHWCCLR	0x0023	35
KRHWRCLR	0x0024	36
SCOLMVTO	0x0025	37
SCOLLNTO	0x0026	38
KRHWCPSS	0x0027	39

Name:	Hex Opcode:	Decimal Opcode:
KRHWGPIX	0x0028	40
KRHWSPIX	0x0029	41
KRHWWFPS	0x002A	42
KRHWWBPS	0x002B	43
KRHWUFPS	0x002C	44
KRHWUBPS	0x002D	45
KRHWSSCOL	0x002E	46
KRHWCPRSS	0x002F	47
KRHWCPXMSS	0x0030	48
KRHWCPYMSS	0x0031	49
LSLUTSL	0x0032	50
LSLUTMD	0x0033	51
LSLUTRD	0x0034	52
LSLUTRS	0x0035	53
LSRLWRRST	0x0036	54
LSRLRDRST	0x0037	55
KRHWSILS	0x0038	56
SCOLSW	0x0039	57
SCOLSESHP	0x003A	58
LSSWLAPAT	0x003B	59
SCOLPLYGN	0x003C	60
SCOLRECT	0x003D	61
SCOLRRECT	0x003E	62
SCOLELLIPSE	0x003F	63
SCOLEARC	0x0040	64
SCOLEPIE	0x0041	65
SCOLESEG	0x0042	66
SCOLHSPLINE	0x0043	67
SCSTHINCR	0x0044	68
SCOLPLYGN	0x0045	69
SCOLSMODE	0x0046	70
SCOLHPEARC	0x0047	71
SCSORG	0x0048	72
SCARRECT	0x004F	79
SCARRRECT	0x0050	80
SCARAPLYGN	0x0051	81

LIB SHELL OPCODES

Name:	Hex Opcode:	Decimal Opcode:
SCARWPLYGN	0x0052	82
SCARELLIPSE	0x0053	83
SCAREPIE	0x0054	84
SCARESEG	0x0055	85
SCARFFILL	0x0056	86
SCARBFILL	0x0057	87
LSDAPAT	0x0058	88
LSSAFAPAT	0x0059	89
LSRAPAT	0x005A	90
SCARTRI	0x005B	91
SCARSPIN	0x005C	92
LSDFONT	0x0065	101
LSSFONT	0x0066	102
LSRFONT	0x0067	103
SCTXSSF	0x0068	104
SCTXSBLROT	0x0069	105
SCTXSCHROT	0x006A	106
SCTXSALIGN	0x006B	107
SCTXSUW	0x006C	108
SCTXSSOW	0x006D	109
SCTXSIM	0x006E	110
SCTXSLM	0x006F	111
SCTXSBF	0x0070	112
SCTXSWPW	0x0071	113
SCTXSTRING	0x0072	114
SCTXSINCS	0x0073	115
SCTXEXTENT	0x0074	116
SCTXSBX	0x0075	117
LSSWPVAR	0x0076	118
SCTXSSW	0x0077	119
SCTXSDMODE	0x0079	121
KRGCFR	0x007E	126
LSDGC	0x007F	127
LSSGC	0x0080	128
LSRGC	0x0081	129
KRGCSDA	0x0082	130

Name:	Hex Opcode:	Decimal Opcode:
KRGCEN	0x0083	131
KRGCDI	0x0084	132
KRGCMV	0x0085	133
KRGCSDM	0x0086	134
LSINQGCP	0x0087	135
V2TXSTRING	0x008E	142
V2GPSWDW	0x008F	143
V2GPSVWP	0x0090	144
V2GPMAP	0x0091	145
V2GPMAPX	0x0092	146
V2GPMAPY	0x0093	147
V2GPUNMP	0x0094	148
V2OLSW	0x0095	149
V2OLMVTO	0x009F	159
V2OLRMVTO	0x00A0	160
V2OLPT	0x00A1	161
V2OLLNTO	0x00A2	162
V2OLRLNTO	0x00A3	163
V2OLPLINE	0x00A4	164
V2OLRPLINE	0x00A5	165
V2OLELLIPSE	0x00A6	166
V2OLEARC	0x00A7	167
V2OLEPIE	0x00A8	168
V2OLESEG	0x00A9	169
V2OLRECT	0x00AA	170
V2OLRRECT	0x00AB	171
V2OLPLYGN	0x00AC	172
V2OLRPLYGN	0x00AD	173
V2OLHPEARC	0x00AE	174
V2ARRECT	0x00B8	184
V2ARRRECT	0x00B9	185
V2ARAPLYGN	0x00BA	186
V2ARARPLYGN	0x00BB	187
V2ARWPLYGN	0x00BC	188
V2ARWRPLYGN	0x00BD	189
V2ARELLIPSE	0x00BE	190
V2AREPIE	0x00BF	191

LIB SHELL OPCODES

Name:	Hex Opcode:	Decimal Opcode:
V2ARESEG	0x00C0	192
V2ARFFILL	0x00C1	193
V2ARBFILL	0x00C2	194
V3GPSHCLIP	0x00CD	205
V3GPSYCLIP	0x00CE	206
V3GPSPROJ	0x00CF	207
V3GPSDIST	0x00D0	208
V3GPSDISTH	0x00D1	209
V3GPSDISTY	0x00D2	210
V3MDID	0x00DE	222
V3MDSET	0x00DF	223
V3MDORG	0x00E0	224
V3MDSC	0x00E1	225
V3MDTRAN	0x00E2	226
V3MDROTX	0x00E3	227
V3MDROTY	0x00E4	228
V3MDROTZ	0x00E5	229
V3VWID	0x00F0	240
V3VWSET	0x00F1	241
V3VWRPT	0x00F2	242
V3VWROTX	0x00F3	243
V3VWROTY	0x00F4	244
V3VWROTZ	0x00F5	245
V3OLMVTO	0x0100	256
V3OLRMVTO	0x0101	257
V3OLPT	0x0102	258
V3OLLNTO	0x0103	259
V3OLRLNTO	0x0104	260
V3OLPLINE	0x0105	261
V3OLRPLINE	0x0106	262
V3OLPLYGN	0x0107	263
V3OLRPLYGN	0x0108	264
V3ARAPLYGN	0x0113	275
V3ARARPLYGN	0x0114	276
V3ARWPLYGN	0x0115	277

Name:	Hex Opcode:	Decimal Opcode:
V3ARWRPLYGN	0x0116	278
LSCPBMZM	0x011E	286
LSBMARMSK	0x011F	287
LSSKIP	0x0120	288
LSWAIT	0x0121	289
LSNOOP	0x0122	290
LSEMULEN†	0x0124	292
LSLUTPM	0x0125	293
LSLUTBLINK	0x0126	294
LSLUTSBLINK	0x0127	295
LSCBM	0x0128	296
LSSBM	0x0129	297
LSFBM	0x012A	298
LSCPBMSM	0x012B	299
LSCPBMMS	0x012C	300
LSRBM	0x012D	301
LSINQSYS	0x012F	303
LSINQ	0x0130	304
LSPKEN	0x0131	305
LSPKDI	0x0132	306
LSPKSIDL	0x0133	307
LSPKSIDH	0x0134	308
LSVSINT	0x0135	309
LSDCLPL	0x0136	310
LSSCLPL	0x0137	311
LSRCLPL	0x0138	312
LSEGASW‡	0x0139	313
LSVGASW‡	0x013B	315
LSTGOBJ	0x0140	320
LSTGLAY	0x0141	321
LSTGROBJ	0x0142	322
LSTGRLAY	0x0143	323
LSTGSLVIS	0x0144	324
LSTGSVTBL	0x0145	325
LSCLCOMPR	0x0148	328

† — PG-Series-1 and PG2-Series-1 only
‡ — PG-Series-1 only

D.2 Opcodes by Command

Name:	Hex Opcode:	Decimal Opcode:
KRGCDI	0x0084	132
KRGCEN	0x0083	131
KRGCFR	0x007E	126
KRGCMV	0x0085	133
KRGCSDA	0x0082	130
KRGCSDM	0x0086	134
KRHWCCLR	0x0023	35
KRHWCPRSS	0x002F	47
KRHWCPSS	0x0027	39
KRHWCPXMSS	0x0030	48
KRHWCPYMSS	0x0031	49
KRHWFCLR	0x0021	33
KRHWGCLS	0x0014	20
KRHWGPIX	0x0028	40
KRHWRCLR	0x0024	36
KRHWRCLS	0x0016	22
KRHWSBCOL	0x001A	26
KRHWSBOP	0x001B	27
KRHWSCLIPR	0x001E	30
KRHWSCLR	0x0022	34
KRHWSCLS	0x0015	21
KRHWSDIRECT	0x0017	23
KRHWSFCM	0x0010	16
KRHWSFCOL	0x0019	25
KRHWSILS	0x0038	56
KRHWSMSK	0x001C	28
KRHWSPIX	0x0029	41
KRHWSSCOL	0x002E	46
KRHWSSMSK	0x001D	29
KRHWSTDEF	0x0018	24
KRHWUBPS	0x002D	45

Name:	Hex Opcode:	Decimal Opcode:
KRHWUFPS	0x002C	44
KRHWWBPS	0x002B	43
KRHWWFPS	0x002A	42
KRIOSSMODE	0x0012	18
LSBMARMSK	0x011F	287
LSCBM	0x0128	296
LSCLAPPEND	0x0007	7
LSCLCLOSE	0x0002	2
LSCLDEL	0x0005	5
LSCLEXEC	0x0003	3
LSCLLOOP	0x0004	4
LSCLMODIFY	0x0008	8
LSCLOPEN	0x0001	1
LSCLREAD	0x0006	6
LSCLREALLOC	0x000A	10
LSCLSBLKSZ	0x0009	9
LSCPBMMS	0x012C	300
LSCPBMSM	0x012B	299
LSCPBMZM	0x011E	286
LSDAPAT	0x0058	88
LSDCLPL	0x0136	310
LSDFONT	0x0065	101
LSDGC	0x007F	127
LSEGASW‡	0x0139	313
LSEMULEN‡	0x0124	292
LSERROR	0x0013	19
LSFBM	0x012A	298
LSINQ	0x0130	304
LSINQGCP	0x0087	135
LSINQMEM	0x000B	11
LSINQSYS	0x012F	303

† — PG-Series-1 and PG2-Series-1 only

‡ — PG-Series-1 only

LIB SHELL OPCODES

Name:	Hex Opcode:	Decimal Opcode:
LSLUTBLINK	0x0125	294
LSLUTMD	0x0033	51
LSLUTPM	0x125	293
LSLUTRD	0x0034	52
LSLUTRS	0x0035	53
LSLUTSBLINK	0x0127	295
LSLUTSL	0x0032	50
LSNOOP	0x0122	290
LSPKDI	0x0132	306
LSPKEN	0x0131	305
LSPKSIDH	0x0134	308
LSPKSIDL	0x0133	307
LSRAPAT	0x005A	90
LSRBM	0x012D	301
LSRCLPL	0x0138	312
LSRDRST	0x0020	32
LSRFONT	0x0067	103
LSRGC	0x0081	129
LSRLRDRST	0x0037	55
LSRLWRRST	0x0036	54
LSSAFAPAT	0x0059	89
LSSBM	0x0129	297
LSSCLPL	0x0137	311
LSSFONT	0x0066	102
LSSGC	0x0080	128
LSSWLAPAT	0x003B	59
LSSWPVAR	0x0076	118
LSVGASW‡	0x013B	315
LSVSINT	0x0135	309
LSWAIT	0x0121	289
LSWRRST	0x001F	31

‡ — PG-Series-1 only

Name:	Hex Opcode:	Decimal Opcode:
SCARAPLYGN	0x0051	81
SCARBFILL	0x0057	87
SCARELLIPSE	0x0053	83
SCAREPIE	0x0054	84
SCARESEG	0x0055	85
SCARFFILL	0x0056	86
SCARRECT	0x004F	79
SCARRRECT	0x0050	80
SCARSPIN	0x005C	92
SCARTRI	0xxx5B	91
SCARWPLYGN	0x0052	82
SCOLEARC	0x0040	64
SCOLELLIPSE	0x003F	63
SCOLEPIE	0x0041	65
SCOLESEG	0x0042	66
SCOLHPEARC	0x0047	71
SCOLHSPLINE	0x0043	67
SCOLLNTO	0x0026	38
SCOLMVTO	0x0025	37
SCOLPLINE	0x003C	60
SCOLPLYGN	0x0045	69
SCOLRECT	0x003D	61
SCOLRRECT	0x003E	62
SCOLSESHP	0x003A	58
SCOLSMODE	0x0046	70
SCOLSW	0x0039	57
SCSORG	0x0048	72
SCSTHINCR	0x0044	68
SCTXEXTENT	0x0074	116
SCTXSALIGN	0x006B	107

LIB SHELL OPCODES

Name:	Hex Opcode:	Decimal Opcode:
SCTXSBF	0x0070	112
SCTXSBLROT	0x0069	105
SCTXSBX	0x0075	117
SCTXSCHROT	0x006A	106
SCTXSIM	0x006E	110
SCTXSINCS	0x0073	115
SCTXSLM	0x006F	111
SCTXSSF	0x0068	104
SCTXSSOW	0x006D	109
SCTXSSW	0x077	119
SCTXSTRING	0x0072	114
SCTXSUW	0x006C	108
SCTXSWPW	0x0071	113
V2ARAPLYGN	0x00BA	186
V2ARARPLYGN	0x00BB	187
V2ARBFILL	0x00C2	194
V2ARELLIPSE	0x00BE	190
V2AREPIE	0x00BF	191
V2ARESEG	0x00C0	192
V2ARFFILL	0x00C1	193
V2ARRECT	0x00B8	184
V2ARRRECT	0x00B9	185
V2ARWPLYGN	0x00BC	188
V2ARWRPLYGN	0x00BD	189
V2GPMAP	0x0091	145
V2GPMAPX	0x0092	146
V2GPMAPY	0x0093	147
V2GPSVWP	0x0090	144
V2GPSWDW	0x008F	143

Name:	Hex Opcode:	Decimal Opcode:
V2GPUNMP	0x0094	148
V2OLEARC	0x00A7	167
V2OLELLIPSE	0x00A6	166
V2OLEPIE	0x00A8	168
V2OLESEG	0x00A9	169
V2OLHPEARC	0x00AE	174
V2OLLNTO	0x00A2	162
V2OLMVTO	0x009F	159
V2OLPLINE	0x00A4	164
V2OLPLYGN	0x00AC	172
V2OLPT	0x00A1	161
V2OLRECT	0x00AA	170
V2OLRLNTO	0x00A3	163
V2OLRMVTO	0x00A0	160
V2OLRPLINE	0x00A5	165
V2OLRPLYGN	0x00AD	173
V2OLRRECT	0x00AB	171
V2OLSW	0x0095	149
V2TXSTRING	0x008E	142
V3ARAPLYGN	0x0113	275
V3ARARPLYGN	0x0114	276
V3ARWPLYGN	0x0115	277
V3ARWRPLYGN	0x0116	278
V3GPSDIST	0x00D0	208
V3GPSDISTH	0x00D1	209
V3GPSDISTY	0x00D2	210
V3GPSHCLIP	0x00CD	205
V3GPSPROJ	0x00CF	207
V3GPSYCLIP	0x00CE	206

LIB SHELL OPCODES

Name:	Hex Opcode:	Decimal Opcode:
V3MDID	0x00DE	222
V3MDORG	0x00E0	224
V3MDROTX	0x00E3	227
V3MDROTY	0x00E4	228
V3MDROTZ	0x00E5	229
V3MDSC	0x00E1	225
V3MDSET	0x00DF	223
V3MDTRAN	0x00E2	226
V3OLLNTO	0x0103	259
V3OLMVTO	0x0100	256
V3OLPLINE	0x0105	261
V3OLPLYGN	0x0107	263
V3OLPT	0x0102	258
V3OLRLNTO	0x0104	260
V3OLRMVTO	0x0101	257
V3OLRPLINE	0x0106	262
V3OLRPLYGN	0x0108	264
V3VWID	0x00F0	240
V3VWROTX	0x00F3	243
V3VWROTY	0x00F4	244
V3VWROTZ	0x00F5	245
V3VWRPT	0x00F2	242
V3VWSET	0x00F1	241

matrox
electronic systems ltd.

1055 ST. REGIS BLVD., DORVAL, QUEBEC H9P 2T4, CANADA
TEL.: (514) 685-2630
TELEX: 05-822798
FAX.: 514-685-2853

Matrox Electronic Systems Ltd. reserves the right to make changes in specifications at any time and without notice. The information furnished by Matrox Electronic Systems in this publication is believed to be accurate and reliable. However, no responsibility is assumed by Matrox Electronic Systems Ltd. for its use, nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Matrox Electronic Systems Ltd.